REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Nov 6, 2000	final 5/24/2000 to 11/6/2000

4. TITLE AND SUBTITLE
F/DOD/ARMY/AMCOM/Common Object Request Broker Architecture (CORBA) Interface R&D

5. FUNDING NUMBERS
DAAH01-98-D-R001 DO 089

6. AUTHORS
Letha Etzkorn Eric Ironside

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Alabama in Huntsville
Research Administration
Huntsville, AL 35899

8. PERFORMING ORGANIZATION REPORT NUMBER
5-21026

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Army Aviation & Missile Command
AMSAM-AC-RD-B
Natalynn Weddle (256) 876-4900
Redstone Arsenal, AL 35898-5380

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Discussed results of a CORBA Interoperability Study, plus the development of a CORBA demonstration software package developed that described the use of CORBA in the Army test environment.

20011105 023

14. SUBJECT TERMS
CORBA

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling each block of the form follow. It is important to ***stay within the lines to meet optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g., 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g., 10 Jul 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C -	Contract	**PR** -	Project
G -	Grant	**TA** -	Task
PE-	Program Element	**WU** -	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with . . . ; Trans. of . . . ; To be published in . . . . When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g., NOFORN, REL, ITAR).

DOD -	See DoDD 5230, "Distribution Statements on Technical Documents"
DOE –	See authorities.
NASA–	See Handbook NHB 2200.2.
NTIS –	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE –	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA–	Leave blank.
NTIS –	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

PLEASE CHECK THE APPROPRIATE BLOCK BELOW

DAO# _____

☐ _____ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☑ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

☐ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTS (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT F:
FUTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

☐ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

☐ This document was previously forwarded to DTIC on _____ (date) and the AD number is _____.

☐ In accordance with provisions of DoD instructions. The document requested is not supplied because:

☐ It will be published at a later date. (Enter approximate date, if known).

☐ Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Letha Etekorn
Print or Type Name

Letha Etekorn
Authorized Signature/Date

(256) 824-6291
Telephone Number

Feasibility Study on the Use of CORBA in the Environment of the U.S. Army Test, Maintenance, and Diagnostic Equipment Activity

Dr. Letha Hughes Etzkorn

Eric Ironside

The University of Alabama in Huntsville

1.0 Introduction

The purpose of this feasibility study (performed under Army contract F/DOD/ARMY/AMCOM/Common Object Request Broker Architecture (CORBA) Interface Research and Development, DAAH01-98-D-R001 DO 089) was to determine if CORBA version 2.2 or higher, was appropriate for current use or future use in the testing software and hardware environment employed by the United States Army Test, Maintenance, and Diagnostic Equipment Activity. The selection of CORBA version 2.2 or higher, rather than an earlier version of CORBA, was required by Joint Technical Architecture-Army [4]:

> "The mandate for distributed object computing is interworking with the Object Management Group (OMG) Object Management Architecture (OMA), composed of the Common Object Request Broker Architecture (CORBA), CORBAservices, and CORBAfacilities."
> "The CORBA interoperability mandate does not preclude the use of other distributed object technologies such as ActiveX/DCOM or Java, as long as the capability for interworking with CORBA application and objects is maintained by the non-CORBA system. Products are available that allow interworking among distributed computing technologies. For CORBA, the following specification is mandated: The Common Object Request Broker: Architecture and Specification, Version 2.2 OMG Group, Inc. OMG document formal, 1 February 1998."

Thus, if distributed objects are utilized within a testing software environment, the capability of interworking with CORBA 2.2 or higher must be provided. Employing CORBA 2.2 (or higher) would be the simplest option (rather than DCOM, or Java Beans) for new systems that provide distributed objects computing, or for older systems being upgraded to distributed object computing.

Within this document, the word "investigators" refers to Dr. Letha Etzkorn, and to Eric Ironside. Dr. Etzkorn is on the faculty of the University of Alabama in Huntsville, and Eric Ironside is a graduate student at the university.

This document is divided as follows: Section 2.0 provides an overview of the TMDE testing, diagnostic, and maintenance environment, as understood by the investigators. Section 3.0 examines other communication options than CORBA that could be used in the TMDE environment. Section 4.0 discusses the CORBA interoperability study performed as part of this contract. Section 5.0 describes a CORBA demonstration that was developed as part of this contract. Section 6.0 provides conclusions. Appendix A contains UML diagrams for the POWER test (selected as a fairly complex and representative sub-test of the overall test suite), Appendix B contains the SPORT ICE DLL documentation, Appendices C through I contain the software developed during the CORBA interoperability study. Appendix J contains the Java version of the CORBA Power test demonstration software. Appendix K contains the C++ version of the CORBA Power test server, for Windows NT. Appendix L contains the C++ version of the CORBA Power test server, for Linux

2.0 TMDE Environment

From the standpoint of the investigators, the TMDE test, diagnostic, and maintenance environment was fairly difficult to understand. Much information about hardware and especially about the software seems to be located in the memories of hardware and software engineers who have worked in development on TMDE projects. Thus, information about the environment does not appear to be systematically documented in any central location. Also, the information seems to be spread across multiple companies and organizations. For example, we talked to several personnel from SAIC, from Litton, and from Oceana, as well as multiple Army employees, in our attempt to glean information about the environment. Most people were generally helpful, although the time they could spend with us was often limited. Due to time constraints we have not talked to all the companies that created the IETM (i.e. RDEC implemented a different version of the Test Manager employed in the IETM).

When developing the UML diagrams that described the tests employed in the STE/ICE system the best information we received was acquired from a) looking at actual code sent

to us by Mr. Tien Chau, and b) information acquired from Mr. Robert Stermon. Mr. Stermon sent us three documents that describe the Litton version of the Interactive Electronic Technical Manual (IETM) and he also examined for correctness multiple UML diagrams that we produced.

We feel at this point that complete documentation of the various systems (IETM, STE/ICE, etc.) could be a lengthy process. We suspect that the best documentation might be derived by going systematically through the source code and providing low level UML diagrams from that. Then the higher level UML diagrams, such as the Use case diagrams, could be developed by talking to various people from various companies to acquire any additional information.. However, as we said earlier, this would seem to be a lengthy process. One problem with the proposed documentation method is that extracting substantial technical information about software from people via vocal discussions such as telephone calls tends to result in incomplete (and sometimes incorrect) information, and is very inefficient in terms of time usage of the people involved. Unfortunately, people seem more likely to provide such information vocally than via email messages or other written documentation.

Our best understanding of the overall TMDE software environment is documented in the structure chart in Figure 1 (Note: all figures are included at the end of this document, but prior to the Appendices). There are 3 versions of the Test Manager shown in this structure chart:

1. Litton version
2. RDEC version
3. MIC/TIM (Message Interface Control/Test Interface Module)

The SPORT ICE DLLs convert from one set of test numbers recognized by the IETM, to a different set of test numbers recognized by SPORT ICE. This was one of the confusion factors in our learning the TMDE environment, in that there seemed to be multiple sets of test numbers. One set of test numbers, those used by the IETM, is published in [7]. The SPORT-ICE DLL test numbers have not been published before, as we understand it. We include a description of the DLL software, provided to us by Robert Starnes, the SPORT-ICE software engineer who developed them, in Appendix B.

Another aspect of the software that surprised us was that, while most of the Graphical User Interface (GUI) is implemented within the IETM, apparently some portions of the GUI are implemented within the SPORT-ICE DLLs. The setup we expected would be to have the SPORT-ICE DLLs return test data in some internal format, and have the IETM format the data for graphical representation.

For the demonstration portion of this project a single test was isolated from the entire test suite contained within the testing environment. Mr. David Zajac selected this test, known as the POWER test, for its sufficient complexity and as a fair representative of the other tests in the testing siute. Figure 2 contains a written description of the POWER test, as we understand it. Appendix A contains the UML diagrams of the POWER test.

3.0 Other Options besides CORBA

There are several communication techniques that, from a technical standpoint, could potentially be used in the TMDE environment. These include:

1. ASCII commands and responses. This is basically the current technique employed.
2. Downloaded HTML pages
3. Common Gateway Interface
4. ActiveX/Distributed Component Object Model (DCOM)
5. Java Beans
6. Common Object Request Broker Architecture (this is the area we focused on primarily in this study)

The ASCII commands and responses technique currently employed by the TMDE has one large advantage in that working software has already been developed. It has multiple drawbacks, however, in regard to its current implementation, and in regard to future expansion to a more general communications environment. As noted above in Section 2.0, the current interfaces are not well documented in a general manner—various documents exist: some have been explicitly released, such as [7], while others have not, to our knowledge, previously been widely released (see Appendix B). Some information about the interfaces is apparently not written down, and is available only through discussion with the engineers responsible for the software and hardware.

Another objection to the current ASCII technique is that it is not easy to expand to a more general communications environment. The current testing situation has one testing

computer connected to one unit under test, such as a HUMVEE. In this point-to-point situation, the current ASCII technique works well. However, in a more general situation, such as one main testing computer connected over a local area network (LAN) to several units under test, with most of the testing being performed by embedded processors on board the units under test, the ASCII technique is a less appropriate communications choice. Any security issues would have to be addressed from scratch—there is no ready-made security handling available. In an even more general situation, where the embedded processors on the units under test communicate with a main testing computer over a mobile communications network, the ASCII technique is even less appropriate. In both of the general situations discussed above, the ASCII would have to be sent over various Ethernet and Internet-related protocols (TCP/IP) (probably using sockets or Remote Procedure Calls), with the use of Mobile IP in the second situation discussed. So the simplicity of ASCII is no longer as appealing.

Downloaded HTML pages are similar to ASCII commands in their transmission and security characteristics. They have one major advantage compared to ASCII in that they can be used with web browsers, so the GUI implementation can be handled largely by the server on the embedded processor within the unit under test, and separate interface software would not be required on the main testing computer.

In the larger, networked situations discussed above, both downloaded HTML pages and ASCII would probably be sent over either sockets or Remote Procedure Calls. Neither of these methods supports objects directly, so either the choice would be made to not use distributed objects, or the objects would have to be implemented as a wrapper around the sockets or RPCs. In many ways, this would be "recreating the wheel," since basically both CORBA and ActiveX/DCOM are object wrappers around Remote Procedure Calls.

The Common Gateway Interface is a standard for interfacing external applications with information servers, such as web servers. A plain HTML document that a web server retrieves is static—a text file that doesn't change. A CGI program can be called in real time, however, to retrieve dynamic information. When using CGI, a web browser communicates with the host server (daemon) using HTTP. When the URL of a CGI program is requested, the CGI services the request, it formats an HTML web page, and returns it to the client web browser via its host server. Thus, with CGI, instead of

getting a series of pre-formatted HTML pages, dynamically-formatted HTML pages are provided, with the HTML pages potentially different with each transmission. CGI can also handle many other types of documents, as well as HTML pages, including both images and audio clips, which expands the potential of a unit under test's embedded testing software. Many different languages, both compiled languages and scripting languages, can be used to write CGI programs, including C++, Perl, and Visual Basic, among others. One major advantage of CGI is that, "unlike Java and other evolving technologies, CGI is a relatively mature protocol. Most Web servers use it." [1].

The primary disadvantage of CGI is the lack of built in security. According to [1], "running a CGI program is like inviting the world to run a program on your system." Because of this, most HTTP servers limit the access of a CGI program to the rest of the system by restricting the directories the CGI program is allowed to access. Thus, if CGI were employed in the non-point-to-point testing scenarios earlier discussed, security would be a major issue that would have to be addressed. CGI does not have any specific security options built in, although it does provide support for transferring authentication information for commonly used authentication procedures to the server.

ActiveX/DCOM is a very versatile solution to distributed object communication. ActiveX has a big advantage over Java-related technologies, in that ActiveX is binary, and downloaded ActiveX controls are able to run at true machine speeds, whereas Java applets are interpreted at runtime, and thus run slower than the machine speed. The primary disadvantages of ActiveX/DCOM are twofold. First, ActiveX/DCOM is primarily a Microsoft technology, mostly implemented on Microsoft platforms. Microsoft has made clear that where the published DCOM differs from the Windows implementation, the Windows implementation should be considered correct [4]. Second, ActiveX controls have full system privileges, so that an ActiveX control can "perform a directory search of the file system, change video resolutions, reboot your computer, or even grab files and send them over the internet to a remote server."[1] Thus, ActiveX has the potential for major security violations, although Microsoft has specified substantial security measures to prevent such violations. These security measures include the use of authentication, trusted domains, workgroups, security Ids, and Access tokens [4]. It

should be noted that the recent versions of the CORBA specification support DCOM interoperability.

The Enterprise JavaBeans specification defines an architecture for a transactional, distributed object system based on components. The specification mandates a programming model and a set of interfaces which make up the EJB API [2]. Enterprise JavaBeans are software components that run in an environment called an EJB container. The EJB container manages an enterprise bean in the same manner that the Java Web Server hosts a servlet or an HTML browser hosts a Java applet. An enterprise bean only funtions inside an EJB container. The EJB container manages all aspects of a Java Bean, including remote access to the bean, security, persistence, transactions, concurrency, and access to and pooling of resources [2].

Java Beans provide a better security model than many other methods, such as CGI. Thus, Java Beans are not allowed access to protected parts of a remote computer. Also, interoperability between systems is inherited from the Java language. These advantages of Java Beans are also its disadvantages—Java Beans cannot currently be implemented in any language other than Java. Thus, legacy code (i.e. Delphi or C++) would have to be manually translated. However, future versions of CORBA (beginning with CORBA 3.0) are required to provide interoperability between CORBA and Java Beans for non-Java applications, so it may be that in the future the Java Beans interface will be very widely used.

The Common Object Request Broker Architecture (CORBA) restricts its usage to the interface between applications and the transport protocol on the local machine. Because of this, communication security is guaranteed by using the correct communication protocols provided by an operating system. This also gives the developer control of how much security is required for the application and what environment the application is developed in (i.e. C++, Delphi, Java, etc. on Windows NT, Windows 2000, Linux, etc.). Finally, CORBA has the added flexibility of being interoperable with ActiveX/DCOM, and, with the recent CORBA 3.0 specification, also provides interoperability with Java Beans (note that currently, as of August, 2000, no CORBA ORBS implement a full CORBA 3.0 specification) [6]. Thus, CORBA as a standard allows the most application flexibility coupled with the most control over system security.

4.0 CORBA Interoperability Study

As part of Army contract F/DOD/ARMY/AMCOM/Common Object Request Broker Architecture (CORBA) Interface Research and Development, DAAH01-98-D-R001 DO 089, a demonstration of CORBA in the TMDE environment was performed by simulating a single standard test. Showing CORBA interoperability, between various operating systems and programming languages, was a desired aspect of this demonstration software. After discussion, the desired operating systems for this demonstration were chosen as Linux and Windows 95/98/NT. The desired programming languages were Java, C++, and Delphi. The initial desired CORBA ORBS were Visibroker (a Borland/Inprise commercial ORB), and the TAO ORB (a well known free academic ORB).

The first phase of implementing this demonstration was to show that the various CORBA ORBs could be connected. We did this by interfacing ORBS with a simple echo interface, in which a string was passed from a client to a server and back to the client. Two slightly different examples of this interface, one employing exceptions, and one without, are shown in Figures 3 and 4. The reason for having two similar interfaces will be explained below.

One of the primary goals for this interoperability study was to examine the use of the CORBA naming service. The CORBA naming service makes it possible for a client to connect to a server without the server's address being passed directly to the client. Rather, the CORBA naming service's address is passed to the client, then the client accesses the server by name, via the naming service. All servers must register, under unique names, with the naming service (it is possible to use multiple naming services...if so, then the same name can be used for different servers in different naming contexts). It was considered that the use of the CORBA naming service could reduce the problem with passing an initial address from a client to a server upon test startup. As it turns out, the naming service approach has limitations. The limitations will be discussed more in Section 5.0 and in the conclusions. From the standpoint of the CORBA interoperability study that was initially performed, the need to employ the CORBA naming service

required an extra level of connectivity. Some ORBS provide a proprietary, non-CORBA compliant, naming service that adds extra functionality in addition to that required by the CORBA naming service specification. This was true of the Visibroker ORB. This was another variation that had to be taken into account during the interoperability study.

4.1 Delphi Visibroker to Delphi Visibroker, on Windows NT

Our first step was to connect from a Delphi/Visibroker client to a Delphi/Visibroker server, using the proprietary Visibroker naming service known as *osagent*. Both client and server were running on Windows NT (as it happens, no Linux version of Delphi currently exists). This was easily performed, using a CORBA Wizard which is provided with the Delphi programming environment. We then expanded the simple example provided by the Delphi Wizard and performed a simplified Delphi to Delphi demonstration of the POWER test. This will be explained further in Section 5.0.

4.2 C++ TAO to C++ TAO on Windows NT

Our second step was to connect from a TAO ORB client on Windows NT version 4.0, to a TAO ORB server, also on Windows NT. We started with the simple connection examples provided with the TAO ORB. None of these examples provided a simple echo interface—although still simple, they were all somewhat more complicated than our chosen connectivity test. Since all of these examples employed the ACE framework, upon which the TAO ORB is based, modification of these examples provided with the TAO ORB required a substantial knowledge of the ACE object-oriented framework. Rather than spend the time to learn what amounted to a new programming language, we decided to access the TAO ORB using standard CORBA calls. Our first example connected a TAO client to a TAO server with the CORBA IOR stringified reference, that provides a CORBA address, of the server object. The IOR was passed explicitly to the client via a file. The code for this example is provided in Appendix C. This example was sub-optimal for our interoperability study since it did not employ the CORBA naming service.

The next connectivity example we performed on the TAO ORB, Windows NT to Windows NT, employed the CORBA-compatible naming service that was provided with the TAO ORB. This example also included exception handling. The Naming Service is run as a separate program, in a separate process. This example worked well, although

we discovered that the version of the TAO ORB that we used had a fault (bug) in relation to the ORB destroy call, which does not work . This was a problem recognized by others and it is further discussed on the TAO mailing list. This fault affected the termination portion of the ORB and had no effect on our implementation of the TAO ORB.

4.3 C++ TAO to C++ TAO on RedHat Linux 6.1, Then to/from C++ TAO on Windows NT

Our next goal was to run the TAO ORB on RedHat Linux version 6.1, from a Linux TAO client to a Linux TAO server. We found that exception handling does not easily work in the TAO ORB on Linux—according to helpers from the TAO mailing list, the compiler used in Linux with the TAO ORB in Linux does not support native exception handling easily and efficiently. There were two different ways to deal with this limitation. One was to implement exception handling as a separate parameter, using special macros provided in the ACE framework. In this case, exceptions had to be included as a special macro inside every parameter list of every function that implemented an exception. The alternative was to use a version of the TAO ORB that did not include exception handling, and to remove all exception handling from the interface and elsewhere. The second option was chosen for this study. The Linux version of the Naming Service example is the same as the Windows NT version, except that all exception handling was removed. (See Figures 3 and 4).

In relation to getting the TAO ORB up and running, there are a couple of things that should be mentioned. Using the default settings provided with the TAO source code, a complete compile of the ACE environment, including the TAO ORB required approximately 6 hours on a Pentium II machine with 128 Megabytes of Ram, and required 2.34 gigabytes of disk storage (the directions provided with the TAO ORB have some hints as to how this could be somewhat reduced; however, this reduction looked fairly difficult). On RedHat Linux 6.1, the TAO ORB required approximately 7 hours to compile on a Pentium III machine with 128 Megabytes of RAM. However, it required only around 500 Megabytes of disk storage. Possibly some parameter (such as Debug) could be set differently in the Windows NT environment to substantially reduce the disk size. Since we were fairly short on time, we did not pursue this reduction. Some pre-compiled (for certain machines and operating systems) versions of the TAO ORB can be

downloaded from a commercial website (commercial support can also be purchased from this website). Note that this large size and time requirements for ORB compilation does NOT affect the final runtime distribution of a CORBA application, rather, it only affects the requirements of the development environment.

After getting the TAO ORB running from Linux client to Linux server, we then connected from a Windows NT client to a Linux server, and from a Linux client to an NT server, all using the TAO ORB, and the TAO Naming service. We did not employ exceptions in this connection. No code modification was required, except that we created a Windows NT version of the echo example, using the CORBA Naming service, that did not include exception handling.

4.4 Java Visibroker Application to Java Visibroker Application, C++ TAO on Windows NT and RedHat Linux

Our next step was to connect from a Java Visibroker client to a Java Visibroker server (using Borland JBuilder), both on Windows NT 4.0. We started with a Java application, as opposed to a Java applet. This code is shown in Appendix E. The following step was to install JBuilder and Visibroker on RedHat Linux 6.1. Then we were able to connect from a Java client application running on Windows NT to a Java server application running on RedHat Linux 6.1 (still using the code shown in Appendix E). Note that this version of the Java echo does not employ a Naming Service. Rather, the stringified IOR reference associated with the servant object is passed to the client via a file. We had to pass this file from a RedHat Linux machine to a Windows NT machine before we could connect.

Our next step was to connect from a Java Visibroker client application to a Java Visibroker server, both on Windows NT 4.0, using the TAO ORB Naming Service. This code is shown in Appendix F. We also got this code to run between a client on Windows NT and a server on RedHat Linux 6.1.

After this we wanted to connect from a TAO client application to a Java Visibroker server, on Windows NT 5.0 and RedHat Linux 6.1, using the Visibroker CORBA-compatible Naming Service (as opposed to the proprietary Visibroker naming service, osagent). The code for the Java Visibroker server is included in Appendix G. This code differs from the server code in Appendix F only in that the root naming service is passed

in as a parameter, rather than included as a stringified IOR reference. We were not able to easily find a way to get the Visibroker CORBA-compatible naming service to produce an IOR stringified reference, which made the Visibroker CORBA-compatible naming service difficult to use with a non-Visibroker client or server. Thus, the new code is:

```
org.omg.CORBA.Object rootObj = orb.resolve_initial_references("NameService");
```

which assumes the initial naming service address is already known. We were able to connect between a Visibroker servant and a TAO client, using Visibroker Naming Service, and between a Visibroker client, TAO servant, using the TAO Naming Service. We were able to run both connections between Visibroker and Windows NT, alternating TAO and Visibroker. However, in this case the Visibroker Client has to use the TAO Naming Service, since it requires the IOR stringified reference of the Naming Service.

4.5 Java Visibroker Applet on Windows NT to C++ TAO on RedHat Linux

Our next plan was to provide a Java applet, as opposed to a Java application. The code for this is provided in Appendix H. In this situation, we used a C++ TAO server on Linux (although we were also able to use the Java Visibroker server, with TAO Naming Service), and accessed an HTML page stored on the Linux machine via Microsoft Internet Explorer. In this case, a Java Visibroker applet was stored on the Linux machine, but was downloaded to Windows NT, and executed in Microsoft Internet Explorer's Java Virtual Machine. This situation worked well. However, we were not successful when replacing Microsoft Internet Explorer with Netscape. Netscape's internal browser is an older version of Visibroker, pre-CORBA 2.2. Microsoft Internet Explorer has no internal CORBA ORB, so we were able to download the Visibroker ORB. With Netscape, it apparently defaults to its internal CORBA ORB. People on the Borland Visibroker newsgroups told us that it is possible to replace the standard Netscape CORBA ORB with another one, by replacing the ORB files as they are stored on the local machine. However, this was not successful for us—we also seemed to be violating some Netscape security aspects (apparently we needed to do signed applets). Due to shortness of time, we did not pursue this further. We later found during implementation of our complete demo, that in a full duplex case, where the downloaded applet included a server as well as a client, that we also ran afoul of Microsoft Internet Explorer internal security (this will be explained more later).

Note the IOR stringified reference stored in the HTML file in Appendix H. This IOR stringified reference has to be set to the IOR stringified reference of the TAO Naming Service, after the Naming Service has started running. If the Naming Service has stopped and been re-started, then the new IOR stringified reference must be copied to this HTML file. However, in this particular situation, the HTML file is local to the Linux server, so this is not a substantial hardship.

4.6 Delphi Visibroker to JavaVisibroker on Windows NT

The Delphi/Visibroker combination does not support the complete CORBA interface; rather, it provides a minimal interface. Delphi's main communication target environment is DCOM, and CORBA is implemented as an option on the DCOM interface (Delphi CORBA code has DCOM-like references). This has the effect of making it difficult to connect to a non-Delphi, non-Visibroker client or server. Until very recently (last spring), the only way that Delphi could be connected to a non-Delphi, non-Visibroker client or server was through either hand-coding a static interface, that is normally produced by a translator program, or by using the Dynamic Invocation Interface. We spent a substantial amount of time trying this, and were not successful. We then tried to connect from a Delphi/Visibroker client to a Java Visibroker servant, using the Visibroker proprietary naming service, osagent. We felt that this was a smaller step, and thus more likely to be successful. We were not able to make our Delphi client find our Java server, even though the equivalent Java client could find it easily. We finally determined that the problem is the Delphi version of Visibroker has not been updated to CORBA 2.2. It is an older version of Visibroker (version 3.4) than the version of Visibroker that is available with either Java or C++ (version 4.0).

Between CORBA 2.1 and CORBA 2.2, the CORBA specification moved from supporting Basic Object Adapters to supporting Portable Object Adapters. The Portable Object Adapters being used in the Java version of Visibroker were simply not visible to the Delphi version of Visibroker. Once Delphi/Visibroker has been upgraded to CORBA 2.2 (or higher—the current version of Visibroker on Java actually implements CORBA 2.3), then the Delphi to Java Visibroker connections should be possible. This also explains why the connections to the TAO ORB were unsuccessful—we were also employing Portable Object Adapters on the TAO ORB.

Although Java Visibroker supports CORBA 2.1 as a separate (backwards compatibility) option, as does the TAO ORB, it does not make sense to go back to using a Basic Object Adapter on Java Visibroker, even though this would presumably have made the Delphi to Java connection possible. The Basic Object Adapter was completely dropped from the CORBA standard in CORBA 2.2, due to problems discovered with it. So the Basic Object Adapter is not CORBA 2.2 compliant, as required by the JTA-A.

Recently, an IDL2PAS compiler has been supplied to be used with the Delphi/Visibroker ORB. This will negate the need for hand-coding static interfaces, and will make the Delphi/Visibroker to other platforms/ORBS much more practical, once it has been upgraded to CORBA 2.2.

4.7 C++ MICO to C++ TAO, C++ MICO to Java Visibroker, Windows NT and RedHat Linux

Since the Delphi/Visibroker combination did not prove interoperable to CORBA 2.2 code (in its current version), we decided to acquire another free ORB to help show CORBA's current interoperability. We chose the MICO ORB, which is a fairly well known academic ORB. The MICO ORB has the philosophy that only what is required by CORBA is implemented, so with MICO there is not a lot of proprietary interfaces, etc., to brush aside, as there was with the TAO ORB and the Visibroker ORB. We fairly easily got C++ MICO to compile and run on both Windows NT and RedHat Linux (we needed some help with the RedHat Linux MICO Makefile, but the MICO mailing list provided us with that help). Afterwards, we very easily made connections to Java/Visibroker, to/from Windows NT and RedHat Linux, and connections to TAO, to/from Windows NT and RedHat Linux. This code is shown in Appendix I.

5.0 Power Test Demonstration

As part of this contract, we were required to provide a demonstration of the potentialities of CORBA in the TMDE environment. Mr. David Zajac decided that an appropriate demonstration of the capabilities of CORBA would be achieved by providing a C++ server running on Linux, and clients written in Delphi and Java. Prior to writing the demonstration software, we performed the CORBA interoperability study earlier described in Section 4.0. During this study, we discovered that Delphi does not currently support CORBA 2.2 or higher. Thus, Delphi was not used in the CORBA demonstration.

We implemented a Java to Java Power Test demonstration first. The code for this is shown in Appendix J. This Power test is implemented as a Java application, rather than a Java applet. We also implemented the client side of the Power test as a Java applet, which runs successfully under the JBuilder applet viewer, but we ran into several security problems when we tried to run this applet in either the Microsoft Internet Explorer browser or the Netscape browser. These security problems arose because the Power test implements a complete bi-directional data transfer, with both a CORBA client and CORBA server on the "client" side application, as well as both a CORBA client and CORBA server on the "server" side application (as described in Section 4, we were earlier able to perform a CORBA connection with a CORBA client running in Microsoft Internet Explorer). The fix for this problem would be to provide signed applets. However, since this appeared to be quite time-consuming, and was not directly related to the purpose of our study, we did not pursue it further.

The Java to Java application, both client side and server side, ran successfully on both Linux and Windows NT.

The next step was to implement a Java client running to a C++ server. We first translated the Power Test Java server into C++ code, running with the TAO ORB under Windows NT. This code is in Appendix K.

We then moved this code to Linux. This was much harder than we anticipated, because we hit a fault, or bug in the TAO ORB. Luckily for us, it was an already documented fault (bug), that had been fixed in a beta version of the ORB. We found this out from the TAO ORB mailing list, which is an excellent resource for anyone using TAO. We downloaded the beta version of the TAO ORB, and compiled it under Linux (this in itself took us a few days—basically, we had to find the right documentation for how to do this. The mailing list was very helpful to us in this, as well). When we compiled the beta version of TAO originally, it supported TAO-specific exceptions, but did not support native exceptions (the binary TAO code for linux that we had used earlier apparently did not support exception handling of any kind). We had a choice of converting our code to employ ACE-specific (ACE is the communications framework upon which TAO is based) exceptions, or re-compiling the ORB to use native exceptions. Native exceptions have poorer performance (according to the TAO mailing list) than ACE-specific

exceptions, when used on Linux, which is why ACE-specific exceptions are the default. However, we spent a few hours trying to use the ACE-specific exceptions in our code, and were not successful. We concluded that using the ACE-specific exceptions would require a more in-depth understanding of the TAO and ACE environment than we currently possessed. So we chose to re-compile the TAO ORB for native exceptions. This was easy, but required a few hours wait for the ORB to re-compile.

Once the beta version of the TAO ORB had been recompiled for native exceptions, our Power Test C++ server easily compiled. We then attempted to run the server on the Linux computer, and attach to it with a Java Visibroker client running on Windows NT. We were initially unsuccessful. The Java Visibroker client kept returning an error that the servant object was not found. In this initial attempt, we were using the new Naming Service that came with the beta version of the TAO ORB. It turned out that the new beta version of the TAO Naming Service was incompatible with the Java Visibroker client. When we went back to an old version of the TAO Naming Service, the Java Visibroker client on Windows NT was easily able to connect to the TAO server on Linux. This was the final variation on the Power Test software interconnectivity that we had been asked to implement. The code for the TAO Linux server, that uses the beta version of the TAO ORB (beta as of 10/27/00) is shown in Appendix L.

6.0 Conclusions

We were successful in implementing a Power Test using CORBA, over multiple languages and operating systems. From this we conclude that CORBA is a potentially useful technique in the TMDE environment, that provides an excellent mechanism for decoupling distributed, interoperable software from a single operating system or programming language. It would provide the advantage of requiring a separate, defined interface, which would be useful in strictly and cleanly defining the TMDE software interface structure. Also, although we did not address security in this study, CORBA provides good security options. Thus, CORBA allows an excellent migration path toward a situation in which a single tester handles multiple "smart" vehicles under test, with each vehicle providing local diagnostic software.

However, in the point-to-point testing case currently used in the TMDE environment, where the soldier hooks up a testing computer to a vehicle and starts the test, CORBA has

a problem in that the stringified IOR address of the CORBA Naming Service (or alternately, the stringified IOR of a CORBA server), which is required to start any CORBA communication, must be passed in some manner prior to the start of the test requiring CORBA communication. This IOR address could not be passed by CORBA itself. It could be passed by use of ASCII transfer, or by the use of CGI, or it could potentially be typed in by the soldier (although the lengthy and cryptic nature of the stringified IOR would tend to prevent the typing option). There is apparently no current alternative to this. Another option would be to use a known domain name. This Interoperable Naming Service (INS) is specified as part of CORBA 3.0. However, it is not currently implemented on all CORBA ORBS [6]. Alternatively, some CORBA ORBS, such as the TAO ORB, implement an automatic way to find their own naming service, usually by internet multicast. However, this is not a CORBA standard, and is totally proprietary to a particular ORB. The TAO ORB also allows you to configure the ORB at compile time to automatically hunt for the naming service of a few other ORBS, but that is also a proprietary feature that is peculiar to the TAO ORB [6].

This problem will presumably be ameliorated somewhat as CORBA 3.0 becomes standard. However, in any case, the use of CORBA seems to be more appropriate to the situation where a single, stationary, tester is connected via a network, (the network itself could be either stationary or mobile), to a number of vehicles, each containing embedded diagnostic software. In this situation, a CORBA naming service could run on the tester, and each vehicle could be configured to automatically connect to it (with either a domain name as in the CORBA INS, or with a stringified IOR reference).

Our overall simple interoperability study was generally successful, although we encountered some security problems when performing a peer-to-peer CORBA connection in which one end was a Java CORBA applet, containing both a CORBA server as well as a CORBA client, in a Microsoft Internet Explorer or Netscape web browser. However, we discovered various problems in porting CORBA software from one ORB to another, from one language to another, and from one operating system to another. Some of these problems resulted from the fact that each ORB has its own way of implementing the standard CORBA calls. (NOTE: Delphi apparently does not currently implement all of the standard CORBA calls). Thus, code cannot normally be transferred from one ORB to

another verbatim, but must always be modified. Porting from one language to another was similar to the usual problems one has when changing languages; however, in a CORBA environment, it is necessary to also consider the change in CORBA language mappings, as well as the language change itself, when performing language translation. We observe that it is fairly difficult to convert from a Java CORBA application to a C++ CORBA application, due to the C++ CORBA mappings required. It is our belief that it would be easier to translate from C++ to Java, since the Java CORBA mappings are easier to handle than the C++ CORBA mappings. Porting from one operating system to another was also not trivial, even when using the same ORB and the same programming language. The C++ ORBS, in particular, operated somewhat differently in the different operating systems we employed (Linux and Windows NT), primarily in regard to exception handling. We had less trouble with the Java ORBS. Another potential problem with CORBA in general use was noted in a CORBA implementation described in a recent dissertation [3] here at the University of Alabama in Huntsville. In this dissertation, an attempt to use CORBA in a simulation environment was described. This attempt was unsuccessful due to the inability to get a CORBA implementation working in Windows NT with the chosen C++ compiler. Not all compilers are supported by all CORBA ORBs. The compiler used in the study was not identified; it was presumably a less well known compiler.

Another problem with CORBA is that the standard, in the recent past, has tended to be modified fairly substantially on a fairly frequent basis. The change from CORBA 2.1 to CORBA 2.2 was a major re-definition of CORBA. The Basic Object Adapter used in earlier versions of CORBA was replaced by the Portable Object Adapter. The Basic Object Adapter, due to problems with its proprietary implementation by various ORB developers, was completely dropped from the CORBA standard. For this reason, CORBA 2.1 software is not easily compatible with CORBA 2.2 software. Many CORBA 2.2 compliant ORBS still contain a Basic Object Adapter, but, as in the case of the Visibroker ORB, may make it difficult to access. The Visibroker ORB changed the libraries that access the Basic Object Adapter, so that code that ran under the CORBA 2.1 compliant Visibroker will not run under the CORBA 2.2 compliant Visibroker without modification.

CORBA 3.0 also contains a major update to CORBA, in that it specifies Java Beans interoperability. This is a big improvement to CORBA interoperability, since the use of Java Beans is becoming widespread. However, it is a major addition to CORBA.

Due to the recent large number of changes in CORBA, an organization such as the TMDE, that wishes to support interoperability between many vendors' ORBS, would be at a disadvantage. The organization would have to periodically choose between staying with an earlier CORBA specification, which might have substantial deficiencies compared to a new specification, or going with the new specification, with which many vendors ORBS might not yet be compatible. In a discussion with Michael Stahl, Siemens' representative to the Object Management Group (the standardization body which handles CORBA specifications), I (Dr. Letha Etzkorn) asked him if there had been any recent CORBA interoperability studies other than the one performed in this study. His reply was that CORBA is currently a moving target, so that ORBS which do not interoperate today may interoperate fully tomorrow; similarly, ORBS which interoperate fully today, may not provide complete interoperability for all aspects of CORBA tomorrow as the ORBS migrate to follow the changing CORBA standard (although hopefully not many changes as substantial as the CORBA 2.1 to CORBA 2.2 change, which dropped the Basic Object Adapter as a requirement, will occur in the future to the CORBA standard. The CORBA 3.0 changes seemed to primarily add additional features to CORBA).

So, currently our recommendation for the TMDE related to CORBA is as follows:

- CORBA is more appropriate when used in a networked environment, than in an environment that is planned to remain point to point. As the TMDE migrates toward a networked environment, CORBA would be an excellent client/server choice, and would provide a standard mechanism for clearly defining software interfaces.
- CORBA has great potentiality with regard to interoperability for the future. Currently, however, any current CORBA implementation would do better choosing one or a small number of CORBA ORBS, in order to retain complete interoperability as the CORBA standard migrates, and as various ORB vendors upgrade their software to follow the changes in the standard.

References

[1] Breedlove,R.F., Anderson, W.F., Barron, B., Bishop, M., Brophy, K., Ferreira, A.M., Hooban, E., Joshi, D.I., Koets, T., Morgan, B., McGregor, R., Oliphant, Z., Sando, S.E., Taylor, D., Tracewell, R., Wainess, R., Web Programming Unleashed, Macmillan Computer Publishing, USA, 1996.

[2] Enterprise JavaBeans Technology Fundamentals Short Course, http://developer.java.sun.com/developer/onlineTraining/EJBIntro/EJBIntro.html, 2000.

[3] Gorman, E., Interfacing Simulations and Knowledge-Based Systems, doctoral dissertation, Computer Science Department, University of Alabama in Huntsville, 2000.

[4] Grimes, R., Professional DCOM Programming, Wrox Press, 1997.

[5] Joint Technical Architecture-Army (JTA-A), Version 6.0, http://arch-odisc4.army.mil/aes/aea/jta-a/jtaa60/html/jtaa60.htm, May 8, 2000.

[6] Stahl, Michael, (head of the Distributed Object Computing group at Siemens, and who is Siemen's representative to the OMG; also an internationally known author on CORBA and various OO technologies), Personal Communication.

[7] STE/ICE-R DESIGN GUIDE for VEHICLE DIAGNOSTIC CONNECTOR ASSEMBLIES , US Army TMDE.

Power Test Description

Note: It appears that there are functions that modify the data being used outside of the power test function, for example (apparently Value[] is a global variable…):

1. ***GetThreshRPM(threshold, *time, step, Data[])*** loops until the RPM is above or below a certain value, then it stores that RPM in Value[0][0] (via Data[]) from test_power and saves the current time using a pointer to t1 or t2 in test_power. Step determines if it waits 20, 2, or 3 seconds (0, 1, 2) before returning an error. **It is within this function that the user is told to accelerate and decelerate.**
2. ***PExecuteTests(0?, Nsamp?, Data[])*** appears to get a value for the RPM with some conditions (I don't know what 0 or Nsamp do…) and stick it into Value[] (via Data[]).

int Test_Power(step)

Step 0 – No Action Performed (Communication Check?)

Step 1 –

A. Set ovcnt to 0;
B. Set up relays for test 12 or 13 returning error if not set properly;
C. For test 12 r_p is equal to idle rev_pulse else it's 1;
D. Verify pCounter is OK; /*???? What is the pCounter????*/
E. Set the Multiplier[0] = r_p * 60,000 and Units[0] = "RPM";

Step 2 – N/A

Step 3 –

A. Get idle governor value (InterValue(0));
B. Check up to 4 times per power test run, that idle governor value is acceptable.
C. If it is acceptable, return 0, else return invalid value (OVRANGE = -12);
D. Note: It's up to the calling function to check this 4 times (ovcnt counts number of iterations within power-test, but there is no loop);

Step 4 –

A. Set thresholds according to vehicle id;
B. Set t2 = current time + 20 seconds;
C. Loop until engine is running, current time > t2, or pExecuteTests returns a fault;
D. Set t1 = current time;
E. Set rpm1 to (rev_pulse/value[0][0])*60,000; /* rev_pulse/value = revolutions/millisecond*/
F. Set value[0][0]=rpm1 and check that rpm1 is not greater than the first threshold RPM;
G. * This is where an auxiliary function says to accelerate *****
H. Wait up to 20 seconds for RPM to be greater than first threshold and record time in t1 when it does;
I. Set t2 = t1;
J. Save the lower threshold RPM at t1 in rpm1;
K. Wait up to 2 seconds for RPM to be greater than second threshold and record time in t2 when it does;
L. Verify that t2 does not equal t1 (resulting in 0 acceleration) and set acceleration equal to (current RPM – rpm1)*1000/(t2-t1); /* 1000 converts (t2-t1) from milli-seconds to seconds */

Step 5 –

A. Set t2 = current time + 2 seconds;
B. Loop until engine is running, current time > t2, or pExecuteTests returns a fault;
C. Set t1 = current time;

D. Set rpm1 to (rev_pulse/value[0][0])*60,000; /* rev_pulse/value = revolutions/millisecond*/
E. Set value[0][0]=rpm1 and check that rpm1 is not less than the second threshold RPM;
F. * This is where an auxiliary function says to decelerate *****
G. Wait up to 3 seconds for RPM to decrease below the second threshold;
H. Set t2 = t1;
I. Save the upper threshold rpm in rpm1;
J. Wait up to 3 seconds for RPM to decrease below the first threshold;
K. Set t2 = current time;
L. Verify that t2 does not equal t1 (resulting in 0 acceleration) and set acceleration equal to first acceleration (from step 4) plus (current RPM – rpm1)*1000/(t2-t1); /* 1000 converts (t2-t1) from milli-seconds to seconds */
M. Save the acceleration and the percent of base if the vehicle has a baseline.
N. * Test is done *****

Figure 2. Power Test Description

```
module echomodule
{
 interface echo
 {
        string echostring(in string message);
 };
};
```

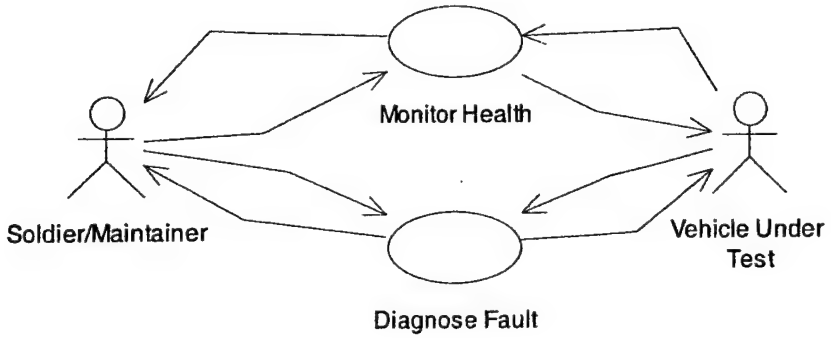
Figure 3. Echo Interface without Exception Handling

```
module echomodule
{

 exception Invalid_Message{ };

 interface echo
 {
        string echostring(in string message)
                raises (Invalid_Message);
 };
};
```

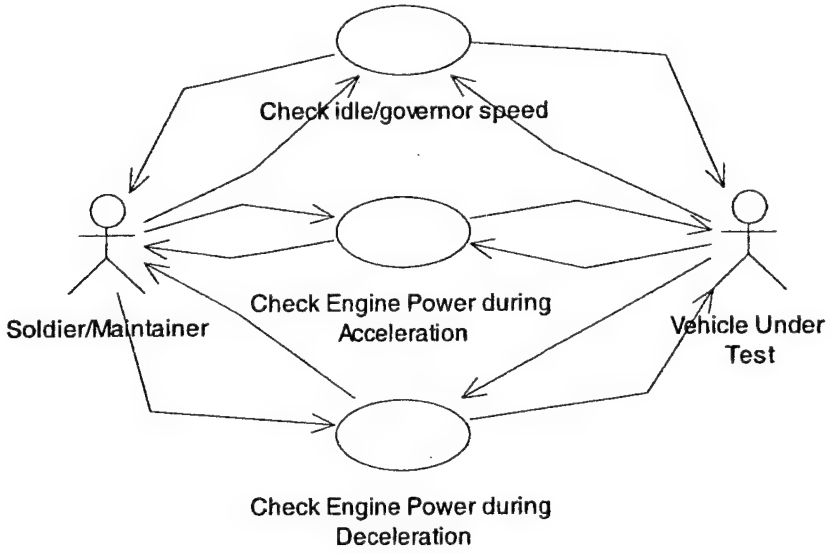
Figure 4. Echo Interface With Exception Handling

Appendix A:

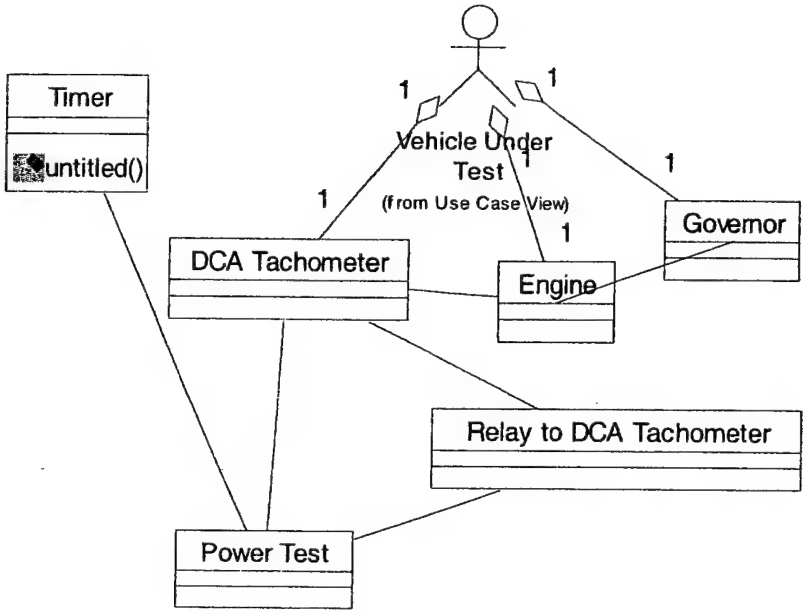
Power Test UML Diagrams



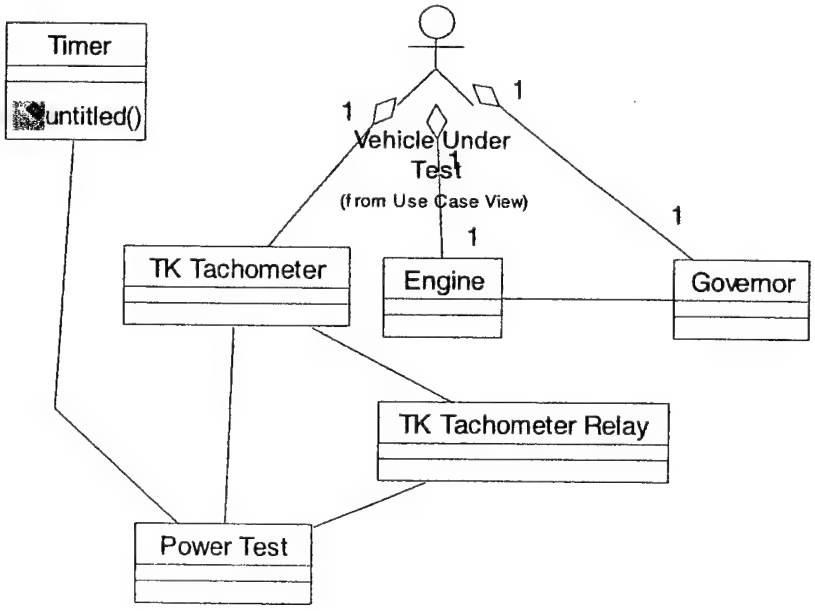
Main Use Case Use Case Diagram: Use Case View



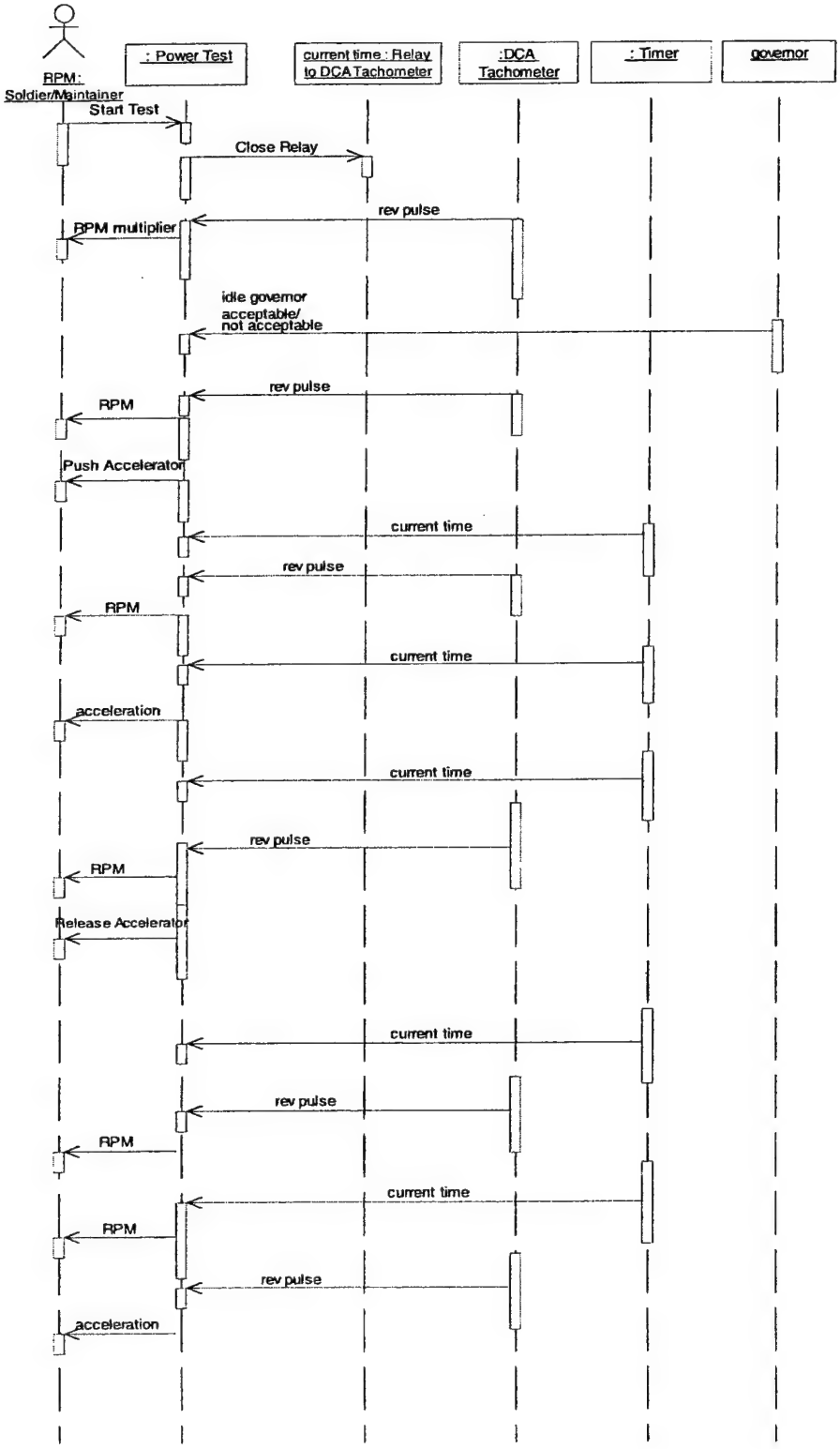
Power Test Use Case Use Case Diagram: Use Case View



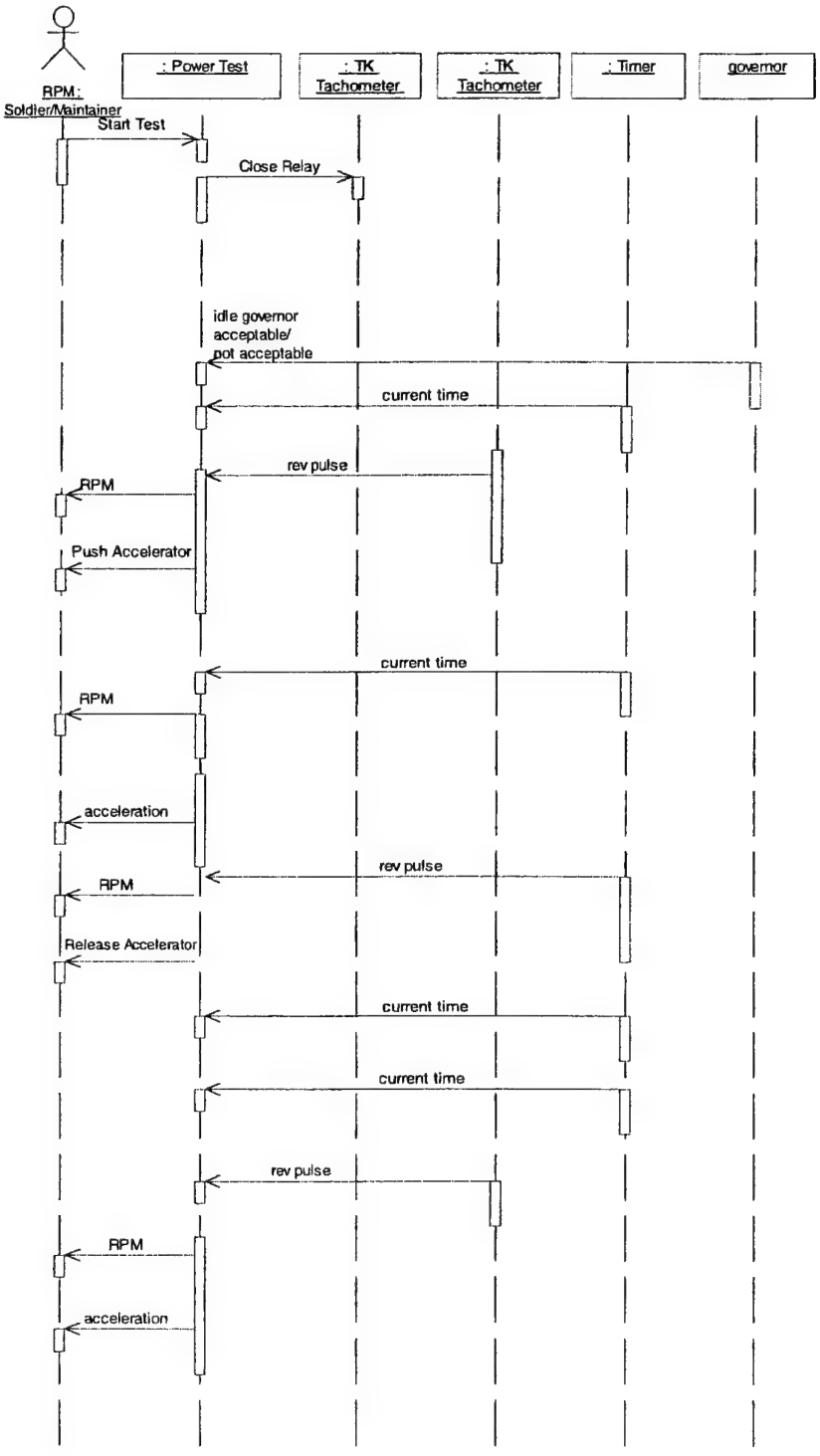
Power Test--DCA Class Diagram: Logical View



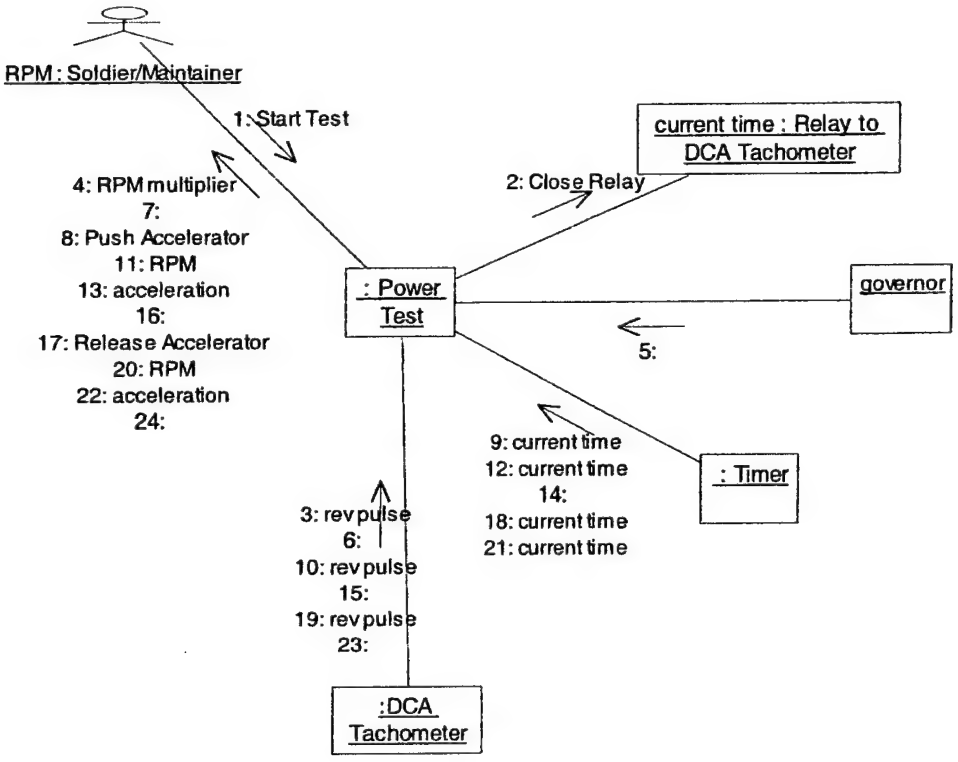
Power Test--TK Class Diagram: Logical View



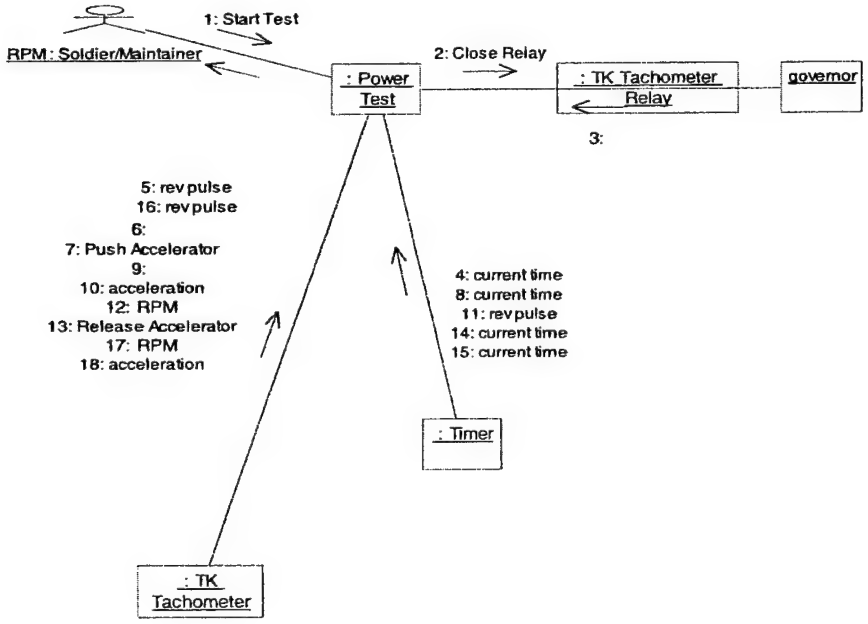
Power Test--DCA Sequence Diagram: Logical View



Power Test--TK Sequence Diagram: Logical View



Power Test--DCA Collaboration Diagram: Logical View



Power Test--TK Collaboration Diagram: Logical View

Appendix B:

SPORT-ICE DLL SOFTWARE INTERFACE

OVERVIEW

The ICE DLLs appear as a 'black box' function which may be communicated with by the calling application only through inputs consisting of the test numbers, the desired mode of execution, and outputs consisting of the test results or error messages. It is the ICE DLLs' responsibility to execute requested tests, display diagrams of tester hardware and any tester specific instructions which may be necessary. The calling applications have no part in, or knowledge of, any ICE tester hardware or anything connected with the execution of the ICE tests.

The SPORT-ICE DLLs have the capability of running the ICE tests in any feasible fashion via the mode selection. The mode determines such things as stopping criteria for looped tests, what is or isn't displayed on the screen by the ICE DLLs, whether results are returned after each iteration of a test loop etc.. This allows a calling application to use the full capabilities of any tester without being involved in or having any knowlege of the actual hardware or test procedures. The modes are generic so that any future testers' software can easily be designed to conform to the same interface spec without any restrictions on its capabilities or efficiency.

ICE TESTS

The ICE tests run by the SPORT-ICE are as follows (NOTE THE CHANGES IN TEST NUMBERS FROM PREVIOUS TESTERS):

** -> test number/name/results change

Test#	Test name
0	Automatic Sequence
10	DCA Engine RPM
11**	TK Engine RPM
12	DCA Power Test
13**	TK Power Test
14	DCA Compression Unbal
15	TK Compression Unbal

21 Oil/Coolant Temperature
22 Temperature
23 Air Cleaner Pressure
24 Fuel Pressure
25 Fuel Supply Pressure
26 Fuel Filter Pressure Drop
27 Fuel Solenoid Voltage
28 Air Cleaner Press Drop
29 Left Air Cleaner Press Drop
30 Turbo Discharge Pressure
31 Left Turbo Dischgarge Pressure
32 Airbox Pressure
35 Engine Oil Pressure
36 Oil Pressure Drop
37 Oil Temperature
38 Coolant Temperature
39 Transmission Pressure
40 Hydraulic Pressure
41 Turbo Pressure
42 AirCleaner Pressure
43 Fuel Pressure
44 Oil Pressure
45 TK Vacuum
46 TK Vacuum Variation 30Hg
47 TK Pressure 50Hg
48 TK Vacuum 150"H2O
49 TK Pressure 25PSI
50 TK Pressure 1000PSI
51 TK Pressure 9999PSI
67 Battery Voltage
68 Starter Motor Voltage
69 Starter Cable Drop
70 Starter Solenoid Volt
71 Average Starter Current
72** Current 1st Peak DCA current/DCA voltage
73** Current 1st Peak TK current/DCA voltage
76** Current 1st Peak DCA current/TK voltage
77** Current 1st Peak TK current/TK voltage
80 Battery Current
81 Electrolyte Level
82 Altenator/Generator Output Voltage
83 Altenator/Generator Field Voltage
84 Altenator/Generator Cable Drop
85 Altenator Output Current Sense
86 Altenator AC Voltage Sense
87 DCA Frequency

88 Live Circuit Low Ohms
89 TK DC Voltage
90 TK DC Current
91 TK Resistance 3K
92 TK Resistance 40K
93 TK AC Voltage
95 TK AC Current
96 TK AC Voltage Frequency
97 TK AC Current Frequency
98 Generic DC Volts
99 Generic AC Voltage

Test number/name changes:

11 = RPM using the TK tachometer (11 previously unused)

13 = Power test using the TK tachometer. Previously, Test 12 was the power test with the result in rpm/sec. Test 13 was the exact same test with the rpm/sec result divided by some constant to get a so called percent which could only be run if the vehicle had so called 'baseline data' associated with it. Now both Test 12 and 13 return the rpm/sec result in Result[0] and if the 'baseline' constant is available the 'percent' result in Result[1].

72, 73, 76, 77:

Choosing any of these test numbers will run the Current 1st peak test whose results include the current 1st peak, battery resistance, starter resistance, and battery resistance change. The four results will be returned in the following order (see description of function RunTest below for details on the Result array):

Result[0] = current 1st peak
[1] = battery resistance
[2] = starter resistance
[3] = battery resistance change

The following tests, due to their nature, are never looped:

Power Tests: 12, 13
Compression Unbalance: 14, 15
Current 1st peak(& associated results): 72, 73, 76, 77
Electrolyte level: 81

All other tests are always looped unless another mode is selected.

Test 0 is an 'autotest' sequence in which all of the DCA tests which can be run on the vehicle are run automatically in sequence according to category and the results are then stored in a file.

The so called 'control functions' used in previous STE-ICE testers are not applicable in SPORT-ICE. Any two tests may be run simultaneously with the following exceptions:

- Tests 12, 13, 14, 15, 72, 73, 76, 77, and 81 can not be run simultaneously with other tests due to their nature.
- AC voltage/current tests may not be run simultaneously with one another
- TK volt/ohm probe tests may not be run simultaneously with one another

The tests results which are displayed are directly selectable by the calling application via the RunTest function argument parameters as explained below.

INTERFACE FUNCTIONS

An application communicates with the SPORT-ICE DLLs using the following four interface functions:

Initialize -initializes the system hardware
InitVehicle - initializes for testing a particular vehicle
RunTest - runs requested tests
StopTest - optional routine used to stop looping tests.

int Initialize(int mode)

Inputs:
mode = INTERNAL, used by the stand alone applications
EXTERNAL, used by external applications such as the IETMs
VADS, used by external applications to explicitly run in VADS mode

Outputs:
0, all OK
err#, if an error occured in initialization

This function 'wakes up' the SPORT-ICE DLLs. This must be the first function called by an application and should not be called again unless a SETUP_ERR, NO_DMM, or PCM_ERR value is returned from a DLL function call.

int InitVehicle(int mode, int vid)

Inputs:
mode = DCAID, only do a DCA ID resistor check
INIT, set the vehicle ID = vid,
if it is the first call to this function, initialize
and calibrate the measurement hardware,
do the vehicle/tester safety checks,
initialize internal variables & flags,

if it's a DCA vehicle, do a DCA ID resistor and DCA sensors check and run offset tests for all DCA sensors present on the vehicle
VID, only update the vid

Returns: DCA ID#, all OK
BAD_DCA, bad DCA ID resistor
BAD_VID, DCA class & vid don't match
err#, if an error occured during self-check (see error list below)

Call this function whenever starting to test a different vehicle or restarting after a severe error has occurred and been corrected, or to only do a DCA ID resistor check. In mode INIT this function initializes and calibrates the hardware if it is the first call to the function in this mode, runs safety checks on the critical vehicle-tester connections, and if the vid identifies the vehicle as a DCA vehicle checks the DCA cable connections, DCA tach indicator, vehicle or tester 12v supply indicator, DCA ID resistor, and determines whether the DCA class indicated by the ID resistor is correct for the vid. If all is OK, a check is made to see which DCA sensors are present on the vehicle. An offset test is run and the results stored for all DCA sensors present in the vehicle. If a valid DCA ID resistor is not detected, only TK tests may be run. There are three circumstances for this case:

The vid belongs to a vehicle known not to have DCA capability.
The measured DCA ID resistor value does not match the range for any DCA class.
The DCA class indicated by the ID resistor does not match the vid.

In the first case a zero will be returned.
In the second case a BAD_DCA error value is returned to inform the caller that only TK tests may be run.
In the third case a BAD_VID error value is returned to inform the caller that only TK tests may be run.
If a valid DCA ID resistor is measured then the dca class corresponding to the resistor value is returned.

If an error other than BAD_DCA or BAD_VID is returned by this function the calling application should not attempt to proceed further.

! NOTE !:

It is recommended that applications call this function in mode DCAID before (or after) each test if the vehicle has DCA capability. This allows the tester to make sure that the DCA cable is still properly connected. If the measured resistor is no longer valid a BAD_VIDDCA error will be returned.

int RunTest(WORD* Test, WORD mode, float* Result, HWND* Hwnd, LPWORD Param)

Inputs:

Test = Pointer to a 2 dimensional array containing the requested test numbers.
If a simultaneous 2nd test is not desired, set Test[1] = 0.
mode = operating mode indicator
Must be set to one of the following main operating modes:
BULKOFF, Used by stand alone mode to run the bulk DCA offsets from InitVehicle.
SETUP, Do a complete setup preparation to run a test(s). Runs an offset test(s) if necessary and prepares everything for the main test(s).
OFFISO, Run an offset only without setting up for the main test. If the offset fails, run the offset fault isolation procedure. All relays are opened on return.
OFFNOISO, Same as OFFSET1 except that the offset fault isolation procedure is not used in cases of failure.
MAINONLY, Run the main test(s) without any setup. RunTest must have already been called in mode SETUP before being called in this mode.
MAIN, Prepare for a test as with mode SETUP, but without running any offset tests. After the setup, run the main test. Display result labels and units with the results.
MAIN2, Same as MAIN except only the results are displayed (no labels or units). This mode is primarily intended for the IETMs which put labels and units around the display boxes themselves.

The following option can be OR'ed with the main mode:
RELAYCLOSED, keep the relays closed after running a main test.

The following stop criterion options are OR'ed with the main mode

for looping tests:
LOOP, loop until the user stops the test (the application calls function StopTest when the user indicates stop)
ITER, Run only one iteration of the test.
TIME, Run the test for the number of milliseconds passed in Param[0]
CRANK, Run for Param[0] msec after the battery voltage drops 0.5v from its initial quiescent value. CRANK & STOP will be displayed to prompt the operator.
START, Same as CRANK except START & STOP prompts will be displayed.
RPM, Run until the engine has been running at > Param[1] rpm for Param[0] msec. ACCEL & DECEL prompts will be displayed.

Result = pointer to a test result array arranged as follows:
For tests other than 12, 13, 14, 15, 72, 73, 76, 77:

Result[0] -> test1 average	Result[4] -> test2 average
[1] -> " minimum	[5] -> " minimum
[2] -> " maximum	[6] -> " maximum

(average = latest running average value)

For Tests 12, 13:
Result[0] = result in 'percent'
Result[1] = result in rpm/sec

For tests 14, 15:
Result[0] = the single result

For Tests 72, 73, 76, & 77:
Result[0] = current 1st peak
[1] = battery resistance
[2] = starter resistance
[3] = battery resistance change

Hwnd = pointer to an array of handles to windows in which the DLL is to display the test values when running in the appropriate modes. Up to three results/test can be displayed simultaneously (Up to four results can be displayed for the Current First Peak test.) One result will be displayed by the DLL in the enclosing rectangle of each window with a non-NULL handle. The Hwnd indices correspond to the Results indices. Hwnd[7] is used by modes CRANK and RPM to display commands as explained above.

Param = pointer to a parameter array containing values used in modes CRANK, START, and RPM as described above.

This function is called to run tests. When called it will perform the following:

Set up the hardware for the test(s) in appropriate modes.
If a hardware setup error occurs, return a negative error code.
If an offset test is run and the result is out of limits, a message to that effect is displayed, the user is given the option to run offset fault isolation, and a failed offset error code is returned if offset fault isolation is not run or is unsuccessful.
In the case of looped tests, the test is run until stopped according to the mode criterion as described above. Whichever of the normal, minimum, or maximum, values that have been chosen by the caller are displayed as the test runs.
On normal completion of a test, a 0 is returned. If an error occurs, a negative error code (from the list below) is returned. For looped tests the values returned in Result are the values at the time the test is stopped.

void StopTest()

This function is used to stopping a looping test. The application should call this function when the appropriate user input for stopping a test is received or the test must be stopped for any other reason.

NOTE: Whenever a looping test which uses the TK volt/ohm probe is run, the SPORT-ICE DLLs automatically provide the option for the user to stop the test using the probe pushbutton. This is handled completely by the DLL.

ERROR CODES

The SPORT-ICE uses the following error codes:

Code	err#	Description
CANCEL	-1	Test cancelled by the user or unacceptable conditions.
BAD_TEST	-2	A test was requested which can not be run on the vehicle being tested. This could happen, for example, if a DCA sensor for the test was not present on the vehicle, or a DCA test was requested and a correct DCA ID resistor had not yet been detected for the vehicle.
OFF_FAIL	-3	The offset test failed.
OVRANGE	-4	The last measured test value exceeds acceptable safety limits.
MEAS_ERR	-5	A hardware error occurred while making a measurement.
INDETERM	-6	The test data is such that a meaningful result can not be obtained
MEM_ERR	-7	Not enough memory available for a requested allocation.
SETUP_ERR	-8	A hardware error occured while setting up for the test
NO_DMM	-9	A hardware measurement device was not detected.
PCM_ERR	-10	An error occured while initializing the measurement hardware.
BATT_LOW	-11	The battery voltage is too low to run the requested test.
TIMEOUT	-12	The test did not complete in the maximum time allowed.

BAD_12V -13: The tester 12v supply voltage is not within specs.
BAD_15V -14: The tester 15v supply voltage is not within specs.
BAD_28V -15: The tester 28v supply voltage is not within specs.
BAD_f_g -16: An improper voltage exists across DCA pins f & g.
BAD_D_g -17: An improper voltage exists across DCA pins D & g.
BAD_U -18: An improper voltage exists across DCA pin U & tester ground.
RELAY_ERR -19: The relay network is not responding to a relay command.
PK_LATE -20: The Current first peak occurred too late to properly compute test results.
BAD_DATA -21: The test data is unfit for analysis.
BAD_DCA -22: The measured DCA ID resistor value is invalid.
BAD_VID -23: The VID entered does not match the detected DCA class.
BAD_TK -24: The measured TK ID resistor value is invalid.
CURR_OVLD -25: The Current First Peak test current is beyond the current sensor's accurate range.
BAD_VIDDCA -26: The DCA resistor does not match the VID
COMM_ERR -30: A communication channel error occurred (applicable only when the VADS is being used).

Appendix C:

C++ TAO ORB Echo Example without CORBA Naming Service

Idl interface (stored in a file called echo.idl):

```
module echomodule
{

 interface echo
 {
        string echostring (in string message);

 };
};
```

client code:

```
#include "echoC.h"
#include "echoS.h"
#include <iostream>

int main (int argc, char* argv[])
{
        //Initialize the ORB
  CORBA::ORB_var orb =
   CORBA::ORB_init (argc, argv,
              "" /* the ORB name, it can be anything! */);

        // Input the IOR of the servant
  if (argc < 2) {
   std::cerr << "Usage: " << argv[0]
         << " client IOR string..." << std::endl;
   return 1;
  }


        CORBA::Object_var obj = orb->string_to_object(argv[1]);
        if (CORBA::is_nil(obj)) {
                        std::cerr << "Nil reference" << std::endl;
                        throw 0;
        }

        echomodule::echo_var myecho;

        try {
                        myecho = echomodule::echo::_narrow(obj);
        } catch (const CORBA::SystemException &se) {
                        std::cerr << "Cannot narrow reference" << std::endl;
                        throw 0;
        }

        std::cout << "Initial message is " << "This is a test" << std::endl;

        CORBA::String_var echostring = myecho->echostring ("This is a test");

        std::cout << "Echoed message is " << echostring.in() << std::endl;
```

```
    // Finally destroy the ORB
    orb->destroy ();

  return 0;
}
```

server code:

```
#include "impl_i.h"
#include "echoS.h"
#include "echoC.h"
#include <iostream>

int main (int argc, char* argv[])
{
 try {
   // First initialize the ORB, that will remove some arguments...
   CORBA::ORB_var orb =
     CORBA::ORB_init (argc, argv,
                      "" /* the ORB name, it can be anything! */);
   CORBA::Object_var poa_object =
     orb->resolve_initial_references ("RootPOA");
   PortableServer::POA_var poa =
     PortableServer::POA::_narrow (poa_object.in ());
   PortableServer::POAManager_var poa_manager =
     poa->the_POAManager ();
   poa_manager->activate ();

          echo_i servant;

          echomodule::echo_var myobject = servant._this();

          CORBA::String_var str=orb->object_to_string(myobject.in());
          std::cout << str.in() << std::endl;

   orb->run ();

   // Destroy the POA, waiting until the destruction terminates
   poa->destroy (1, 1);
   orb->destroy ();
 }
 catch (CORBA::Exception &ex) {
   std::cerr << "In server, CORBA exception raised!" << std::endl;
 }
 return 0;
}
```

File impl_i.h:

```
#include "QuoterS.h"
#include <string>

class echo_i : public POA_echomodule::echo {
public:
  echo_i ();

  char *echostring (const char * message);
```

```
private:
};
```

File impl_i.cpp:

```
#include "Stock_i.h"
#include <iostream>

echo_i::echo_i ()

{
}

char *
echo_i::echostring (const char * message)
{
        std::cout << message << std::endl;

        char * junkie=CORBA::string_dup(message);
        return junkie;

}
```

Appendix D:

C++ TAO ORB Echo Example With CORBA Naming Service

Idl interface (stored in a file called echo.idl):

```
module echomodule
{

 exception Invalid_Message{};

 interface echo
 {
        string echostring(in string message)
                raises (Invalid_Message);
 };
};
```

client code:

```
#include "echoC.h"
#include "orbsvcs\orbsvcs\CosNamingC.h"
#include <iostream>

int main (int argc, char* argv[])
{
 try {
  // First initialize the ORB, that will remove some arguments...
  CORBA::ORB_var orb =
   CORBA::ORB_init (argc, argv,
                "" /* the ORB name, it can be anything! */);


  CORBA::Object_var naming_context_object = orb->string_to_object(argv[1]);

  CosNaming::NamingContext_var naming_context =
   CosNaming::NamingContext::_narrow (naming_context_object.in ());

  CosNaming::Name name (1);
  name.length (1);
  name[0].id = CORBA::string_dup ("echo");

  CORBA::Object_var obj =
   naming_context->resolve (name);

  // Now downcast the object reference to the appropriate type

        echomodule::echo_var myecho;
        try {
                myecho=echomodule::echo::_narrow(obj);
        } catch (const CORBA::SystemException &se) {
                std::cerr << "Cannot narrow reference" << std::endl;
                throw 0;
        }

        std::cout << "Initial message is " << "This is a test" << std::endl;
```

```
        CORBA::String_var echostring = myecho->echostring("This is a test");

        std::cout << "Echoed message is " << echostring.in() << std::endl;


        //NOTE: There's a bug in this code if the ORB is destroyed.  I think it has
        //   to do with
        //somehow destroying some _var variables illegally (that's just a guess).  It
        // apparently
        //has NOTHING to do with the data transmission.  This bug was also in the
        //downloaded TAO examples .  It was NOT added by me.
  // Finally destroy the ORB
 //  orb->destroy ();
 }
 catch (CORBA::Exception &) {
  std::cerr << "CORBA exception raised!" << std::endl;
 }
 return 0;
}
```

server code:

```
#include "impl_i.h"
#include "orbsvcs\orbsvcs\CosNamingC.h"
#include <iostream>

int main (int argc, char* argv[])
{
 try {
  // First initialize the ORB, that will remove some arguments...
  CORBA::ORB_var orb =
    CORBA::ORB_init (argc, argv,
                "" /* the ORB name, it can be anything! */);
  CORBA::Object_var poa_object =
   orb->resolve_initial_references ("RootPOA");
  PortableServer::POA_var poa =
   PortableServer::POA::_narrow (poa_object.in ());
  PortableServer::POAManager_var poa_manager =
   poa->the_POAManager ();
  poa_manager->activate ();

  // Create the servant
        echo_i servant;

  // Activate it to obtain the object reference
        echomodule::echo_var myobject=servant._this();

  // Get the Naming Context reference
  CORBA::Object_var naming_context_object =
   orb->resolve_initial_references ("NameService");
  CosNaming::NamingContext_var naming_context =
   CosNaming::NamingContext::_narrow (naming_context_object.in ());

  // Create and initialize the name.
  CosNaming::Name name (1);
  name.length (1);
```

```
    name[0].id = CORBA::string_dup ("echo");

    // Bind the object
    naming_context->bind (name, myobject.in ());

    orb->run ();

    // Destroy the POA, waiting until the destruction terminates
    poa->destroy (1, 1);
    orb->destroy ();
   }
   catch (CORBA::Exception &) {
    std::cerr << "CORBA exception raised!" << std::endl;
   }
   return 0;
}
```

File impl_i.h:

```
#include "echoS.h"
#include <string>

class echo_i : public POA_echomodule::echo {
public:
  echo_i ();


  char *echostring (const char * message)
            throw (echomodule::Invalid_Message);

private:
};
```

File impl_i.cpp:

```
#include "impl_i.h"
#include <iostream>

echo_i::echo_i ()

{
}

char *
echo_i::echostring(const char * message) throw (CORBA::SystemException)
{
        std::cout << message << std::endl;

        char * junkie=CORBA::string_dup(message);
        return junkie;
}
```

Appendix E:

Java Visibroker Echo Example without CORBA Naming Service

Idl interface (stored in a file called echomodule.idl):

```
module echomodule {
 interface echo {
  string echostring(in string message);
 };
};
```

client code:

```
import org.omg.CORBA.*;

public class Client {

 public static void main(String[] args) {
  // Initialize the ORB.
  org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

  org.omg.CORBA.Object obj = orb.string_to_object(args[0]);
  echomodule.echo echo = echomodule.echoHelper.narrow(obj);

  String mystring = "This is a test";
  System.out.println("The value to be sent is "+mystring);
  String echoed_string = echo.echostring(mystring);

  // Print out the balance.
  System.out.println
   ("The echoed value is " + echoed_string);
 }

}
```

server code:

```
import org.omg.PortableServer.*;

public class Server {

 public static void main(String[] args) {
  try {

   // Initialize the ORB.
   org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

   // get a reference to the root POA
   POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
   // Create policies for our persistent POA
   org.omg.CORBA.Policy[] policies = {
    rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),

rootPOA.create_request_processing_policy(RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
   };
   // Create myPOA with the right policies
   POA myPOA = rootPOA.create_POA( "echo_poa", rootPOA.the_POAManager(),
                policies );
```

```
    // Create the servant

    echoImpl myServant = new echoImpl();

    myPOA.set_servant(myServant);
    org.omg.CORBA.Object ref;

    // Decide on the ID for the servant
    byte[] myservantId = "echoModuleecho".getBytes();
    // Activate the servant with the ID on myPOA
   myPOA.activate_object_with_id(myservantId, myServant);

    // Activate the POA manager
    rootPOA.the_POAManager().activate();


    ref = myPOA.create_reference_with_id("echo".getBytes(),
    "IDL:echomodule/echo:1.0");

    System.out.println(orb.object_to_string(ref));

    // Wait for incoming requests
    orb.run();
   }
   catch (Exception e) {
    e.printStackTrace();
   }
  }
}
```

Appendix F:

Java Visibroker Echo Example With TAO CORBA Naming Service

IDL Interface (stored in a module called echomodule.idl):

```
module echomodule {
 interface echo {
  string echostring(in string message);
 };
};
```

client code:

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class Client {

 public static void main(String[] args) {
  try {
    // Initialize the ORB.
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

    // Obtain the root context.
    org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);

    NamingContext root = NamingContextHelper.narrow(rootObj);

    NameComponent[] qs_name = new NameComponent[1];
    qs_name[0]= new NameComponent("echo","");
    org.omg.CORBA.Object echoObj=root.resolve(qs_name);
    echomodule.echo echo = echomodule.echoHelper.narrow(echoObj);

    String mystring = "This is a test";
    System.out.println("The value to be sent is "+mystring);
    String echoed_string = echo.echostring(mystring);
    System.out.println
     ("The echoed value is " + echoed_string);
  }
  catch (Exception e) {
   e.printStackTrace();
  }
 }
}
```

server code:

```
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
public class Server {
 public static void main(String[] args) {
  try {
   // Initialize the ORB.
   org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
   // get a reference to the root POA
```

```
    POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

    // get a reference to the Naming Service root_context
  org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
  NamingContext root = NamingContextHelper.narrow(rootObj);

    // Create policies for our persistent POA
    org.omg.CORBA.Policy[] policies = {
     rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),
     rootPOA.create_request_processing_policy(
          RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
    };
    // Create myPOA with the right policies
    POA myPOA = rootPOA.create_POA( "echo_poa",
                                                    rootPOA.the_POAManager(),
                            policies );
    // Create the servant
    echoImpl myServant = new echoImpl();

    myPOA.set_servant(myServant);
    org.omg.CORBA.Object ref;

    // Decide on the ID for the servant
    byte[] myservantId = "echo".getBytes();
    // Activate the servant with the ID on myPOA
    myPOA.activate_object_with_id(myservantId, myServant);

    // Activate the POA manager
    rootPOA.the_POAManager().activate();

    myPOA.servant_to_reference(myServant));
    NameComponent [] qs_name = new NameComponent[1];
    qs_name[0] = new NameComponent("echo","");

    ((NamingContext)root).bind(qs_name, myPOA.servant_to_reference(myServant));
    orb.run();
   }
   catch (Exception e) {
    e.printStackTrace();
   }
  }
}
```

Instructions for Running this Example:
Visibroker to Visibroker using TAO Naming Service
Do the following:

start up the TAO Naming Service. Use:
Naming_Service -o ior_output_file

Then copy the ior_output_file from the server directory to the client directory. Copy it into a file in the server directory called kk.bat (in Linux,should be an executable file), and also into another file in the client directory called jj.bat. Modify kk.bat (using Notepad or vi, for example), to have "vbj Server" before the stringified IOR reference in the file. Modify jj.bat to have "vbj Client" (note: don't put in the quotes) before the stringified IOR reference in the file.

Then you can start the Visibroker server by typing kk.bat in the server directory.

When you type jj.bat in the client directory, you should get an echoed response from the Visibroker client via the Visibroker server. Note that in this case you are doing Visibroker to Visibroker, but using the TAO Naming Service.

Appendix G:

Java Visibroker Echo Example With Visibroker CORBA-compatible Naming Service (NOTE: does not use the Visibroker proprietary naming service, osagent)

IDL Interface (stored in a module called echomodule.idl):

```
module echomodule {
 interface echo {
  string echostring(in string message);
 };
};
```

server code:

```
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.io.*;

public class Server {

 public static void main(String[] args) {
  try {
   // Initialize the ORB.
   org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
   // get a reference to the root POA
   POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

   // get a reference to the Naming Service root_context
   org.omg.CORBA.Object rootObj = orb.resolve_initial_references("NameService");

   DataOutputStream dos = null;
   dos= new DataOutputStream(new FileOutputStream("ior_output_string"));
   dos.writeBytes(orb.object_to_string(rootObj));
   dos.close();

   NamingContextExt root = NamingContextExtHelper.narrow(rootObj);

   // Create policies for our persistent POA
   org.omg.CORBA.Policy[] policies = {
    rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),
    rootPOA.create_request_processing_policy(
         RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
   };
   // Create myPOA with the right policies
   POA myPOA = rootPOA.create_POA( "echo_poa",
        rootPOA.the_POAManager(),
                        policies );
   // Create the servant

   echoImpl myServant = new echoImpl();

   myPOA.set_servant(myServant);
   org.omg.CORBA.Object ref;

   // Decide on the ID for the servant
   byte[] myservantId = "echo".getBytes();
```

```
    // Activate the servant with the ID on myPOA
    myPOA.activate_object_with_id(myservantId, myServant);

    // Activate the POA manager
    rootPOA.the_POAManager().activate();

//    ref = myPOA.create_reference_with_id("echo".getBytes(),
//    "IDL:echomodule/echo:1.0");

   // Associate the bank manager with the name at the root context
    // Note that casting is needed as a workaround for a JDK 1.1.x bug.
    ((NamingContext)root).bind(root.to_name("echo"), myPOA.servant_to_reference(myServant));

    orb.run();
   }
   catch (Exception e) {
    e.printStackTrace();
   }
  }
}
```

Instructions for Running this Example:

There are two directories involved in this effort, NamingEcho3 containing the Java Visibroker code, and CompatibleNamingService, containing the C++ TAO code.

a) Visibroker servant, TAO client, using Visibroker Naming Service
start the naming server (in the NamingEcho3 directory) by typing:
start nameserv NameService &

Now start the Visibroker servant in the NamingEcho3 directory by typing:
start vbj -DSVCnameroot=NameService Server

Now copy the ior_output_string file that was produced by the Server to the TAO directory, CompatibleNamingService.

Now, in the TAO directory, CompatibleNamingService, type:
client file://ior_output_string

you should get echoes

b) Visibroker client, TAO servant, using TAO Naming Service

start up the TAO Naming Service. Use:
Naming_Service -o ior_output_file

Now start the TAO servant in the CompatibleNamingService directory. Type:
server

Copy the ior_output_file to the NamingEcho3 directory. Use it to make a jj.bat file as before (containing vbj Client IOR...)

Type jj.bat to run the Client. You should get echoed values.

Appendix H:

Java Visibroker Echo Applet Using TAO CORBA Naming Service

IDL Interface (stored in a module called echomodule.idl):

```
module echomodule {
 interface echo {
  string echostring(in string message);
 };
};
```

Html file to access the applet:

```
<h1>Java IOR Applet Demo</h1>
<hr>
<left>
   <applet archive="vbjorb.jar" code="clientapp.class" width="800" height="100">
     <param name="org.omg.CORBA.ORBClass" value="com.inprise.vbroker.orb.ORB">
     <param name=IOR
value=IOR:000000000000001849444c3a6563686f6d6f64756c652f6563686f3a312e300000000001000000
0000000044000102000000000e3134362e3232392e322e313838000418000000240050 4d43000000040000
00102f62616e6b5f6167656e745f706f6100000000046563686f00000000>
     <h2>You are probably not running a java-enabled browser.
     Please us a Java-enabled browser (or enable your
     browser jor Java) to view this applet...</h2>
   </applet>
</center>
<hr>
```

client code:

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.awt.*;
import javax.swing.*;

public class clientapp extends java.applet.Applet {
GridLayout gridLayout1 = new GridLayout();
 private TextField _echostring, _sendstring;
 private Button _send;
 private echomodule.echo echo;
 private Label _label1, _label2, _label3;

 public void init() {

  gridLayout1.setColumns(2);
  gridLayout1.setRows(3);
  this.add(_label1 = new Label("Echoed String"));
  this.add(_echostring = new TextField());
  this.add(_label2 = new Label("Send String"));
  this.add(_sendstring = new TextField());
  this.add(_label3 = new Label(""));
  this.add(_send = new Button("Send"));
  _echostring.setEditable(false);
  _echostring.setSize(300,20);
  _sendstring.setSize(300,20);
```

```
    try {
      // Initialize the ORB.
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(this,null);

       // Obtain the root context.
      String ref = getParameter ("IOR");
      _echostring.setText(ref);
      org.omg.CORBA.Object rootObj = orb.string_to_object(ref/*args[0]*/);


    echo = echomodule.echoHelper.narrow(rootObj);


      String mystring = "Eric Says Hello!";
      //System.out.println(
            // "The value to be sent is "+mystring);

      String echoed_string = echo.echostring(mystring);

      //System.out.println
       //("The echoed value is " + echoed_string);
      updateecho("Client Started");

    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  public String updateecho(String mystring){
    _echostring.setText("Waiting for echo");
    String echoed_string = echo.echostring(mystring);
    _echostring.setText("Got an Echo");
    _echostring.setText(echoed_string);
    return(echoed_string);
  }

  public boolean action(Event event, java.lang.Object obj) {
      java.lang.Object oTarget = event.target;
      if(oTarget instanceof Button)
      {  String mystring = _sendstring.getText();
        //System.out.println(
                    //          "The value to be sent is "+ mystring);
        _sendstring.setText("Value Sent");
        String echoed_string = updateecho(mystring);
        //System.out.println(
                    //"The echo recieved is "+echoed_string);
        _sendstring.setText("");
        return true;
      }
      return false;
    }
  }
```

Appendix I:

C++ MICO ORB to Java Visibroker, C++ TAO, on Windows NT and RedHat Linux

IDL Interface (stored in a module called echo.idl):

```
module echomodule {
 interface echo {
  string echostring(in string message);
 };
};
```

client code:

```
#include <iostream.h>
#include <iomanip.h>
#include "echo.h"

#include <mico/CosNaming.h>

//--------------------------------------------------------------------------

int
main(int argc, char * argv[])
{
  try
  {
        // Initialize orb
        CORBA::ORB_var orb = CORBA::ORB_init (argc, argv);

        CORBA::Object_var nobj;
        nobj = orb->resolve_initial_references ("NameService");

        CosNaming::NamingContext_var inc;
        inc = CosNaming::NamingContext::_narrow (nobj);

        CosNaming::Name name;
        name.length (1);
        name[0].id = CORBA::string_dup("echo");
        //name[0].kind = "";

        CORBA::Object_var obj;

        try
        {
          obj = inc->resolve (name);
        }
        catch (const CosNaming::NamingContext::NotFound &)
        {
          cerr << "No name found" << endl;
          throw;
        }
        catch (const CORBA::Exception & e)
        {
          cerr << "Resolve failed: " << e << endl;
          throw;
        }
```

```
        if (CORBA::is_nil (obj))
        {
          cerr << "Nil reference for alex_server" << endl;
          throw;
        }

        // Narrow
        echomodule::echo_var tm;
        try
        {
          tm = echomodule::echo::_narrow (obj);
        }
        catch (const CORBA::Exception & e)
        {
          cerr << "Cannot narrow reference: " << e << endl;
          throw;
        }

        if (CORBA::is_nil (tm))
        {
          cerr << "Reference has wrong type" << endl;
          throw;
        }

        // Write its stringified reference to stdout
        CORBA::String_var str = orb->object_to_string (tm);
        cout << str << endl;

    cout << "Message to be echoed is " << "This is a test \n";
    char * tod = tm->echostring("This is a test");

        cout << "Echoed string is " << tod << "\n";
        //cout << "Char is " << tod.in() << "\n";
}
  catch (const CORBA::Exception & e)
  {
    cerr << "CORBA exception: " << e << endl;
    return 1;
  }
  catch (...) {
        cerr << "Error ?" << endl;
    return 1;
  }
 return 0;
}
```

server code:

```
include <time.h>
#include <iostream.h>
#include <stdio.h>
#include "server.h"
#include <mico/CosNaming.h>

char *
echo_impl::
echostring(const char *message) throw(CORBA::SystemException)
```

```
{

  char *junkie= CORBA::string_dup(message);
  return junkie;

}

int
main(int argc, char * argv[])
{
  try
  {
    // Initialize orb
    CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);

    // Get reference to Root POA.
    CORBA::Object_var obj
      = orb->resolve_initial_references("RootPOA");
    PortableServer::POA_var poa
      = PortableServer::POA::_narrow(obj);

    // Activate POA manager
    PortableServer::POAManager_var mgr
      = poa->the_POAManager();
    mgr->activate();

    // Create an object
    echo_impl time_servant;

    // Write its stringified reference to stdout
    echomodule::echo_var tm = time_servant._this();
    CORBA::String_var str = orb->object_to_string(tm);
    cout << str << endl;

        // Register server with Naming Service (nsd)
        {
          CORBA::Object_var obj;
          obj = orb->resolve_initial_references ("NameService");

          CosNaming::NamingContext_var inc;
          inc = CosNaming::NamingContext::_narrow (obj);

          CosNaming::Name name;

          name.length (1);
          name[0].id = CORBA::string_dup("echo");

          inc->bind (name, tm);
        }

    // Accept requests
    orb->run();
  }
  catch (const CORBA::Exception & e)
  {
    cerr << "CORBA exception: " << e << endl;
```

```
    return 1;
  }
  return 0;
}
```

Instructions for Compiling and Running this Example, MICO to MICO on RedHat Linux (Windows NT instructions are very similar):

mico-c++ -c *.cc
mico-ld echo.o server.o -o server -lmico2.3.2 -lmicocoss2.3.2
mico-ld echo.o client.o -o client -lmico2.3.2 -lmicocoss2.3.2

Start the executables as follows:

nsd -ORBIIOPAddr inet:localhost:1234 &
server -ORBNamingAddr inet:localhost:1234 &
client -ORBNamingAddr inet:localhost:1234
Alternately, to make the MICO Naming Service produce a stringified IOR,

nsd --ior /tmp/nsd.ior

Using an Interoperable Naming Service (INS) feature you can then start your client the following way:

client -ORBInitRef NameService=file:///tmp/nsd.ior

This is the "portable way" to pass the IOR to the client. Another way would be to use -ORBNamingIOR. However, this is a MICO specific option.

Appendix J:

CORBA Power Test Demonstration Software, Java Version, for both Windows NT and Linux

vehiclemodule.idl:

```
module VehicleModule{
  interface vehicleServer{
    long test_power(in short step);
    void set_running(in boolean running);
    void set_acc(in short acc);
    boolean set_baseline(in float acc);
    short get_index();
    boolean add_client(in string ior, in string name);
  };

  interface vehicleClient{
    void set_rpm(in short rpm);
    void set_testparm(in string parm);
    void results(in float accel, in float pcf);
  };
};
```

sserver.java:

```
import javax.swing.UIManager;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;

public class SServer implements Runnable{

 //static public PowerTest myServant;
 static public Thread pthread;
 static public Thread cthread;
 static public String[] args;


 public void run(){
  //Start up CORBA ORB
   try{
     //Initialize the orb
     org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
     System.out.println("Got to A");
     //Get a reference to the root POA
     POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

     //Get a reference to the Naming Service root_context
     org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
     NamingContext root = NamingContextHelper.narrow(rootObj);
     System.out.println("Got to B");
     //Create policies for our Persistant POA
```

```
    org.omg.CORBA.Policy[] policies = {
      rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),

rootPOA.create_request_processing_policy(RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
    };

    //Create the POA with above policies
    POA myPOA = rootPOA.create_POA("Vehicle_POA",
    rootPOA.the_POAManager(), policies);
    System.out.println("Got to C");
    //Create the Servant
    SPowerTest myServant = new SPowerTest();
    myPOA.set_servant(myServant);
    org.omg.CORBA.Object ref;
    System.out.println("Got to D");
    //Decide on the ID for the servant
    byte[] myservantId = "Vehicle".getBytes();
    System.out.println("Got to E");
    //Activate the POA Manager
    myPOA.activate_object_with_id(myservantId, myServant);
    System.out.println("Got to F");
    //Activate the POA Manager
    rootPOA.the_POAManager().activate();
    System.out.println("Got to G");
    NameComponent [] qs_name = new NameComponent[1];
    qs_name[0] = new NameComponent("Vehicle","");
    System.out.println("Got to H");

    ((NamingContext)root).bind(qs_name,myPOA.servant_to_reference(myServant));
        System.out.println("Got to I");
    orb.run();

  }
  catch (Exception e){
    e.printStackTrace();
  }

 }

 //Main method
 public static void main(String[] a) {
  args = a;

  SServer n = new SServer();
  cthread = new Thread(n);
  System.out.println("Starting Corba Thread");
  cthread.start();
  System.out.println("Corba Thread is runninng...");

 }
}
```

spowertest.java:

```
import java.util.*;
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;
import javax.swing.UIManager;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.io.*;

public class SPowerTest extends VehicleModule.vehicleServerPOA
{
 static final int OVRANGE = -4;
 static final int TIMEOUT = -12;
 static final int INDETERM = -6;
 static final int BAD_DATA = -21;
 static int ovcnt,
        clnum,          // The number of clients bound to server
        ienum;          // The number of IETMs bound to server
 static float rpm1,
          rpm2,         // Added to hold second RPM value
          pcf,          // Added to hold percentage value
          baseline = -1; // Added to held baseline acceleration
 static float thresh1,      // Lower RPM Threshold
          thresh2,      // Upper RPM Threshold
          accel,        // Acceleration value
          r_p;          // Not used in demo
 long t1,          // Timer used in accel calc
    t2;          // Timer used in accel calc
 //Get the GUI for the server
 static public VehicleFrame frame = new VehicleFrame();
 static cnode clist = null;
 static String [] args;

 SPowerTest(){

   System.out.println("Setting up frame...");
   frame.RPM.setValue(0);
   frame.Accelerator.setValue(0);
   frame.setSize(250,300);
   frame.setVisible(true);
   System.out.println("Finished setting up frame...");
 }

 private int GetThreshRPM(int thresh, long time, int step)
 {
  if(step == 0)
  {  System.out.println("Starting 20sec test. CurrTime=" + System.currentTimeMillis() +
    " time = " + time);
    this.update_clients(true,"Accelerate",false,0,0);
    //check for 20 seconds for rpm to exceed thresh
    while(System.currentTimeMillis() < (time+20000))
    {   if(frame.RPM.getValue()>thresh)
        {
          time = System.currentTimeMillis();
          return 0;
```

```
        }
      }
      System.out.println("Didn't make it through 20 sec thresh test");
    }
    else if(step == 1)
    {  //check for 2 seconds for rpm to exceed thresh
      while(System.currentTimeMillis() < (time+2000))
      {   if(frame.RPM.getValue()>thresh)
          {
            time = System.currentTimeMillis();
            t2 = time;
            return 0;
          }
      }
      System.out.println("Didn't make it through 2 sec thresh test");
    }
    else if(step == 2)
    {  this.update_clients(true,"Decelerate",false,0,0);
      //check for 3 seconds for rpm to come below thresh
      while(System.currentTimeMillis() < (time+3000))
      {   if(frame.RPM.getValue()<thresh)
          {
            time = System.currentTimeMillis();
            return 0;
          }
      }
      System.out.println("Didn't make it through 3 sec thresh test");
    }

    return BAD_DATA;
  }

  public int test_power(short step)
  {
     /***Test_Power*********************************************************
        Measures CI engine power.
        Test 12 uses the DCA tachometer.
        Test 13 uses the TK tachometer.

        If step = 0, Do nothing
           1, Set up hardware & parameters for the test.
           3, Compute the latest test result while the test is looping during
              the idle/governor speed check,
           4, Run the acceleration portion of the test.
           5, Run the deceleration portion of the test & finish up.

        Returns:  0 -> test OK
                 -n -> error n
     ***********************************************************************
     Notes:
      Removed lines that called outside functions not available to the demo
      (i.e. relay setpoints).
      Removed lines that used global variables not pertinent to demo (i.e. DATA)
      Changed function GetTickCount to system time function
      moved static variables into class description
      changed ovcnt from WORD to int
```

```
        Changed Value to Current RPM
        Added rpm2
        ************************************************************************/

       int ret;

//        float *Data[2];

// The following added in place of corba functions
      int tstnum[] = {12,0};
      String Units = new String("");

// Use Data to store past results
//        Data[0] = Value[0];
//        Data[1] = Value[1];

     System.out.println("Received Request " + step);
        if(step == 0) {
      for(int i = 0; i < 1000000000; ++i); // Delay added for netowrk syncro
      return 0;
     }

        else if(step == 1) // Set up for the idle/governor check and/or main test:
        {  // Demo: Initialize all vars
      rpm1 = 0;
      rpm2 = 0;       // Added to hold second RPM value
      pcf = 0;        // Added to hold percentage value
      thresh1 = 0;     // Lower RPM Threshold
      thresh2 = 0;     // Upper RPM Threshold
      accel = 0;       // Acceleration value
      t1 = 0;         // Timer used in accel calc
      t2 = 0;         // Timer used in accel calc
      ovcnt = 0;
        if(tstnum[0] == 12)//add get_tstnum function to corba

// Next 2 lines removed for demo
        {   //if( (ret = pRelay(c_d, 0, CT1F)) < 0) return ret;
        //r_p = rev_pulse;
        }
        else
// Next 2 lines removed for demo
     {   //if( (ret = pRelay(TKrelay[0], 0, CT1F)) < 0) return ret;
              //r_p = 1.;
        }

// Next 2 lines removed for demo
//        if( (ret = pCounter(0, 1, CG_1M)) < 0) return ret;
//        Mult[0] = r_p * 6.0e4;


     Units = "rpm"; // add set_units to corba
        return 0;
      }

      else if(step == 3) // Compute idle, governor check running test value:
      {   ret = 0; // Originally set to InterValue(0);
```

```
// Deglitch spikes which get thru tach filter:
    if(ret == OVRANGE)
    {  //Deglitch
   if(++ovcnt < 4) ret = 0;
}
    else ovcnt = 0;
    return ret;
 }

 else if(step == 4) // Run the acceleration portion of the test:
 {  // Prepare for Power test:
    if(frame.vid == 21 || frame.vid == 23)
    {  thresh1 = 2000;
          thresh2 = 3000;
    }
    else
    {  thresh1 = 1000;
          thresh2 = 2000;
    }

// Make sure engine is still running and get start time & rpm:
    t2 = (System.currentTimeMillis() + 20000); // 20 sec to get started
System.out.println("Thresh set. Current Time: " + System.currentTimeMillis() + ". T2 = " + t2);
    t1 = System.currentTimeMillis();
while(frame.RPM.getValue() <= 0)
    {  //Following removed for demo
   //if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;
   t1 = System.currentTimeMillis();
   System.out.println("t1 = " + t1);
          if( t1 > t2) return TIMEOUT;
    }
System.out.println("Made it past first RPM check. Curr Time ="+System.currentTimeMillis() + "
t1=" + t1 + " t2=" + t2);
    rpm1 = frame.RPM.getValue();//(rev_pulse * 6.0e4)/Value[0][0];

// Must start with the rpm < thresh1:
    //if( (Value[0][0] = rpm1) > thresh1) return INDETERM;
if( rpm1 > thresh1) return INDETERM;

// Loop until rpm > thresh1 or 20 sec has elapsed:
System.out.println("Going into first thresh check. rpm1:" + rpm1 + " Time = " + t1);
    if( (ret = GetThreshRPM((int)thresh1, t1, 0)) < 0) return ret;
System.out.println("made it past first threshhold test. Curr Time ="+System.currentTimeMillis());
t1 = System.currentTimeMillis();
    t2 = t1;
//Modified rpm1 to rpm2     //save low thresh t
    rpm2 = frame.RPM.getValue();//Value[0][0];//low thresh rpm
System.out.println("Made it past second RPM check("+ rpm2 +"). Curr Time
="+System.currentTimeMillis());

// Loop until rpm > thresh2 or 2 sec has elapsed:
System.out.println("Going into 2nd thresh check t2="+t2+". Curr Time
="+System.currentTimeMillis());
    if( (ret = GetThreshRPM((int)thresh2, t2, 1)) < 0) return ret;
```

```
// Compute acceleration result:
    if( (t2 -= t1) == 0) accel = 0;
    else accel = (float)((thresh2 - thresh1) * 1000.)/t2;
System.out.println("Got first accel="+accel+". Curr Time
="+System.currentTimeMillis());
    return 0;
  }

  else if(step == 5) // Run the deceleration portion of the test:
  {  // Make sure engine is still running with rpm > thresh2 & get start time & rpm:
System.out.println("Starting 5. Curr Time ="+System.currentTimeMillis());
t2 = (System.currentTimeMillis()) + 2000;
    //for(;;)
t1 = System.currentTimeMillis();
System.out.println("Getting first RPM. Curr Time
="+System.currentTimeMillis());
while(frame.RPM.getValue()<thresh2)
    {  //Removed for demo
   //if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;
            System.out.println("t1="+t1+". Curr Time
        ="+System.currentTimeMillis());
   if( (t1 = (System.currentTimeMillis())) > t2) return TIMEOUT;
    }
    rpm1 = frame.RPM.getValue();//(rev_pulse * 6.0e4)/Value[0][0];
    //if( (Value[0][0] = rpm1) < thresh2) return INDETERM;
System.out.println("Got first rpm. Curr Time ="+System.currentTimeMillis());
if( rpm1 < thresh2) return INDETERM;
System.out.println("First RPM OK. Curr Time ="+System.currentTimeMillis());
// Loop until rpm < thresh2 or 3 sec has elapsed:
System.out.println("Getting first thresh time. Curr Time
="+System.currentTimeMillis());
    if( (ret = GetThreshRPM((int)thresh2, t1, 2)) < 0) return ret;
System.out.println("Got first thresh. Curr Time ="+System.currentTimeMillis());
t1 = System.currentTimeMillis();
    t2 = t1;
    //rpm1 = Value[0][0];
rpm1 = frame.RPM.getValue();

// Loop until rpm < thresh1 or 3 sec has elapsed:
System.out.println("Getting second thresh. Curr Time
  ="+System.currentTimeMillis());
    if( (ret = GetThreshRPM((int)thresh1, t2, 2)) < 0) return ret;
System.out.println("Got second thresh. Curr Time
="+System.currentTimeMillis());
t2 = System.currentTimeMillis();


// Compute deceleration result:
    if( (t2 -= t1) == 0) return BAD_DATA;
System.out.println("Data is good. Curr Time ="+System.currentTimeMillis());
    //accel += ((rpm1 - Value[0][0]) * 1000.)/t2;
    accel += ((thresh2 - thresh1) * 1000.)/t2;
accel = accel/2;
System.out.println("Accel = " + accel + ". Curr Time
="+System.currentTimeMillis());
```

```
            // Finish up:
            // Next 3 lines modified for demo
                //Value[0][0] = accel;
                //if(VIDdata[vid].pcf) Value[0][1] = accel/VIDdata[vid].pcf;
                //else Value[0][1] = 0;

            if(baseline >0) // If there is baseline data
              pcf = accel/baseline;
            System.out.println("Updating Clients. Curr Time
            ="+System.currentTimeMillis());
            this.update_clients(true,"",true,accel,pcf);


// Next line removed for demo
//              LoadDisplayBuff(4, 0, 2);
             }
             return 0;
 }

 public void set_running(boolean running){
   System.out.println("Set_running Called ..." + running);
   if(running == true)
   {  frame.LabelOnOff.setText("ON");
      System.out.println("Setting vehicle on " + frame.RPM.getValue());
      frame.RPM.setValue(frame.Accelerator.getValue() + 650);
      frame.jLabel4.setText("RPM = " + frame.RPM.getValue());
      frame.StartStop.setText("Stop Vehicle");
   }
   else
   {  frame.LabelOnOff.setText("OFF");
      System.out.println("Setting vehicle off " + frame.RPM.getValue());
      frame.StartStop.setText("Start Vehicle");
      frame.RPM.setValue(0);
      frame.jLabel4.setText("RPM = " + frame.RPM.getValue());
   }
   update_clients();
}

public void set_acc(short acc){
  System.out.println("Set_acc Called...");
  if(frame.LabelOnOff.getText() == "ON")
  {  frame.Accelerator.setValue(acc);
     frame.RPM.setValue(frame.Accelerator.getValue()*65+650);
     frame.jLabel4.setText("RPM = " + frame.RPM.getValue());
  }
  else
  {  frame.RPM.setValue(0);
     frame.jLabel4.setText("RPM = " + frame.RPM.getValue());
  }
  update_clients();
  //clist.vclient.set_rpm((short)frame.RPM.getValue());

}

public boolean set_baseline(float acc){
  System.out.println("Set_baseline Called");
```

```
    if(acc>0)
    {  baseline = acc;
       return true;
    }
    return false;
  }

  public short get_index(){
   return (short)(clnum+1);
  }

public boolean add_client(String ior, String name)
  {   VehicleModule.vehicleClient client;
     try{
      //Initialize the ORB
      System.out.println("Adding Client with ior " + ior);
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

      System.out.println("Client ORB initialized...");
      //obtain the root context
      org.omg.CORBA.Object rootObj = orb.string_to_object(ior);
      System.out.println("Converted Client IOR...");
      client = VehicleModule.vehicleClientHelper.narrow(rootObj);
      System.out.println("Found Client " + name + " ...");
      if(clist == null)
      {   System.out.println("Client List is Empty ...");
         clist = new cnode();
         System.out.println("Created new node ...");
         clist.vclient = client;
         System.out.println("Client " + clnum + " object set");
         clist.name = name;
         System.out.println("Client " + clnum + " name set to " + name);
      }
      else
      {  System.out.println("Set up a new node...");
        cnode temp = new cnode();  // Create a new node
        System.out.println("Set client to new node...");
        temp.vclient = client;     // Add client to new node
        System.out.println("Set name to new node...");
        temp.name = name;          // Add client name to new node

        System.out.println("Create end pointer...");
        cnode end = clist;         // Create temp node
        System.out.println("Walk through list...");
        while(end.next != null)    // walk through list
          end = end.next;

        System.out.println("Store temp node at end of list...");
        end.next = temp;           // Add temp to end of list
      }

      // Determine what type of client it is...
      if(name.startsWith("v"))
      {   clnum++;
         if(clnum == 1) return true;
      }
```

```
      else
      {  ienum++;
         if(ienum == 1) return true;
      }
    }
    catch (Exception e){
      e.printStackTrace();
    }
    return false;
  }
  void update_clients(boolean up, String p, boolean ur, float a, float pcf){
     int i = 0;
     cnode tback = null;
     cnode temp = clist;

     while(temp != null)
     {  try{
          temp.vclient.set_rpm((short)frame.RPM.getValue());
          if(temp.name.startsWith("I") && up)
             temp.vclient.set_testparm(p);
          if(temp.name.startsWith("I") && ur)
             temp.vclient.results(a,pcf);
        }
        catch(Exception e){
          if(temp.name.startsWith("v"))
             --clnum;
          else
             --ienum;

          if(tback != null){
            tback.next = temp.next;
            temp = null;
            temp = tback;
          }
          else{
            clist = temp.next;
            temp = null;
            temp = clist;
            continue;
          }
        }
        tback = temp;
        temp = temp.next;
        ++i;
     }
  }

  void update_clients(){
    int i = 0;
    cnode tback = null;
    cnode temp = clist;

    while(temp != null)
    {  try{
         temp.vclient.set_rpm((short)frame.RPM.getValue());
       }
```

```
      catch(Exception e){
        if(temp.name.startsWith("v"))
          --clnum;
        else
          --ienum;

        if(tback != null){
          tback.next = temp.next;
          temp = null;
          temp = tback;
        }
        else{
          clist = temp.next;
          temp = null;
          temp = clist;
          continue;
        }
      }
      tback = temp;
      temp = temp.next;
      ++i;
    }
  }
}

class cnode{
  VehicleModule.vehicleClient vclient;
  cnode next;
  String name;
  cnode(){
    vclient = null;
    next = null;
    name = "";
  }
}

class VehicleFrame extends JFrame {
  static int vid = 123433;
  //static VehicleModule.vehicleServer vehicle;
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel5 = new JLabel();
  JLabel LabelOnOff = new JLabel();
  JSlider Accelerator = new JSlider();
  JProgressBar RPM = new JProgressBar();
  JToggleButton StartStop = new JToggleButton();
  JPanel jPanel1 = new JPanel();

  public VehicleFrame(/*String[] args*/) {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
```

```
    }


  }

  private void jbInit() throws Exception {
    this.setSize(250,300);
    jLabel1.setText("Accelerator");
    jLabel1.setBounds(new Rectangle(9, 5, 74, 19));
    this.getContentPane().setLayout(null);
    jLabel2.setText("Maximum");
    jLabel2.setBounds(new Rectangle(83, 25, 60, 19));
    jLabel3.setText("Minimum");
    jLabel3.setBounds(new Rectangle(78, 188, 60, 19));
    jLabel4.setText("RPM = 0");
    jLabel4.setBounds(new Rectangle(140, 4, 257, 20));
    jLabel5.setText("The Vehicle Is ");
    jLabel5.setBounds(new Rectangle(57, 211, 83, 27));
    LabelOnOff.setText("OFF");
    LabelOnOff.setBounds(new Rectangle(140, 211, 101, 27));
    Accelerator.setValue(0);
    Accelerator.setOrientation(JSlider.VERTICAL);
    Accelerator.setSnapToTicks(true);
    Accelerator.setMajorTickSpacing(25);
    Accelerator.setPaintTicks(true);
    Accelerator.setMinorTickSpacing(10);
    Accelerator.setPaintLabels(true);
    Accelerator.setToolTipText("");
    Accelerator.setBorder(BorderFactory.createRaisedBevelBorder());
    Accelerator.setBounds(new Rectangle(7, 23, 70, 184));
    Accelerator.setEnabled(false);
    RPM.setOrientation(JProgressBar.VERTICAL);
    RPM.setMaximum(7500);
    RPM.setStringPainted(true);
    RPM.setBounds(new Rectangle(10, 8, 50, 164));
    StartStop.setText("Start The Vehicle");
    StartStop.setBounds(new Rectangle(9, 232, 195, 30));
    StartStop.setEnabled(false);
    jPanel1.setBorder(BorderFactory.createRaisedBevelBorder());
    jPanel1.setBounds(new Rectangle(141, 25, 70, 181));
    jPanel1.setLayout(null);
    this.setResizable(false);
    this.setTitle("Vehicle Server");
    this.getContentPane().add(jLabel1, null);
    this.getContentPane().add(Accelerator, null);
    this.getContentPane().add(jLabel2, null);
    this.getContentPane().add(jLabel3, null);
    this.getContentPane().add(jLabel4, null);
    this.getContentPane().add(jLabel5, null);
    this.getContentPane().add(LabelOnOff, null);
    this.getContentPane().add(StartStop, null);
    this.getContentPane().add(jPanel1, null);
    jPanel1.add(RPM, null);
  }
}
```

cclient.java:

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;
import javax.swing.UIManager;
import javax.swing.UIManager;
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.beans.*;
import javax.swing.UIManager;


public class CClient {
  static VehicleModule.vehicleServer vehicle;
  static String  [] args;
  static String cname = "vehicleClient";
  static int index = 0;
  static boolean en;


  //Construct the application
  CClient(String [] a) {
     args = a;
     try{
      //Initialize the ORB
      System.out.println("Starting CORBA for client client");
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

      System.out.println("CC ORB Initialized");
      //obtain the root context
      org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
      System.out.println("CC Naming converted...");
      NamingContext root = NamingContextHelper.narrow(rootObj);
      System.out.println("CC Nameing narrowed...");
      //Locate an account manager through the naming service
      NameComponent[] qs_name = new NameComponent[1];
      qs_name[0] = new NameComponent("Vehicle","");
      org.omg.CORBA.Object vehicleObj = root.resolve(qs_name);
      vehicle = VehicleModule.vehicleServerHelper.narrow(vehicleObj);
```

```
    index = vehicle.get_index();
    System.out.println("Client #" + index + " is ready");
    cname += index;
    }
    catch (Exception e){
      e.printStackTrace();
    }

  //Start up CORBA ORB
   try{
    //Initialize the orb
    org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
    System.out.println("Client Server ORB initialized");
    //Get a reference to the root POA
    POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

    //Get a reference to the Naming Service root_context
    org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
    NamingContext root = NamingContextHelper.narrow(rootObj);
    System.out.println("CS Naming Narrowed...");
    //Create policies for our Persistant POA
    org.omg.CORBA.Policy[] policies = {
      rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),

rootPOA.create_request_processing_policy(RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
    };

    //Create the POA with above policies
    POA myPOA = rootPOA.create_POA(cname+"_POA",
    rootPOA.the_POAManager(), policies);
    System.out.println("CS POA Created ...");
    //Create the Servant
    CVehicle myServant = new CVehicle(vehicle);
    myPOA.set_servant(myServant);
    org.omg.CORBA.Object ref;
    System.out.println("CC Servant initialized...");
    //Decide on the ID for the servant
    byte[] myservantId = cname.getBytes();
    System.out.println("CC Servant ID gotten...");
    //Activate the POA Manager
    myPOA.activate_object_with_id(myservantId, myServant);
    ref = myPOA.create_reference_with_id(cname.getBytes(),
        "IDL:VehicleModule/" + cname + ":1.0");
    System.out.println("CC Activate object...");
    //Activate the POA Manager
    rootPOA.the_POAManager().activate();
    System.out.println("CC POA Manger activate...");
    NameComponent [] qs_name = new NameComponent[1];
    qs_name[0] = new NameComponent(cname,"");
    System.out.println("CC Name Component initialized...");

    ((NamingContext)root).bind(qs_name,myPOA.servant_to_reference(myServant));
    System.out.println("CC Context bound.... Running ORB");

    System.out.println("CC registering client...");
    en = vehicle.add_client(orb.object_to_string(ref),cname);
```

```
      //Enabling Client doesnt work
       System.out.println("CC calling enabled... Enabled = " + en);
       myServant.set_enable(en);
       System.out.println("CC enabled set...");


       System.out.println("CC Running ORB");
       orb.run();

     }
     catch (Exception e){
       e.printStackTrace();
     }
    }


   //Main method
   public static void main(String[] a) {
     System.out.println("Starting Client Main");
     try {
       UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
     }
     catch(Exception e) {
       e.printStackTrace();
     }
     new CClient(a);
   }
}


class CVehicle extends VehicleModule.vehicleClientPOA
{
  static VehicleModule.vehicleServer vehicle;
  static CGUI frame;
  boolean enabled = true;

  CVehicle(VehicleModule.vehicleServer v){
    vehicle = v;
    frame = new CGUI(vehicle);
    frame.setVisible(true);
  }

  public void set_enable(boolean en){
    System.out.println("got to CVehicle, enabling...En = " + en);
    enabled = en;
    System.out.println("Going to CGUI...Enabled = " + enabled);
    frame.set_enable(true);
    System.out.println("Going to CGUI...Enabled = " + enabled);
    frame.set_enable(enabled);
  }

public void set_rpm(short rpm){
    if(rpm>0)
    {   frame.LabelOnOff.setText("ON");
        frame.StartStop.setText("Stop Vehicle");
    }
```

```
      else
      {   frame.LabelOnOff.setText("OFF");
          frame.StartStop.setText("Start Vehicle");
      }
      frame.RPM.setValue((int)rpm);
      frame.Accelerator.setValue(((int)rpm-650)/65);
      frame.jLabel4.setText("RPM = " + rpm);
    }

    public void set_testparm(String parm){

    }

    public void results(float accel, float pcf){

    }
}

class CGUI extends JFrame{
  static int vid = 123433;
  static VehicleModule.vehicleServer vehicle;
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel5 = new JLabel();
  JLabel LabelOnOff = new JLabel();
  JSlider Accelerator = new JSlider();
  JProgressBar RPM = new JProgressBar();
  JToggleButton StartStop = new JToggleButton();
  JPanel jPanel1 = new JPanel();
  static boolean enabled;


  CGUI(VehicleModule.vehicleServer v) {
    vehicle = v;

    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }

  }

  private void jbInit() throws Exception {
    this.setSize(250,300);
    jLabel1.setText("Accelerator");
    jLabel1.setBounds(new Rectangle(9, 5, 74, 19));
    this.getContentPane().setLayout(null);
    jLabel2.setText("Maximum");
    jLabel2.setBounds(new Rectangle(83, 25, 60, 19));
    jLabel3.setText("Minimum");
    jLabel3.setBounds(new Rectangle(78, 188, 60, 19));
    jLabel4.setText("RPM = 0");
```

```
 jLabel4.setBounds(new Rectangle(140, 4, 257, 20));
 jLabel5.setText("The Vehicle Is ");
 jLabel5.setBounds(new Rectangle(57, 211, 83, 27));
 LabelOnOff.setText("OFF");
 LabelOnOff.setBounds(new Rectangle(140, 211, 101, 27));
 Accelerator.setValue(0);
 Accelerator.setOrientation(JSlider.VERTICAL);
 Accelerator.setSnapToTicks(true);
 Accelerator.setMajorTickSpacing(25);
 Accelerator.setPaintTicks(true);
 Accelerator.setMinorTickSpacing(5);
 Accelerator.setPaintLabels(true);
 Accelerator.setToolTipText("");
 Accelerator.setBorder(BorderFactory.createRaisedBevelBorder());
 Accelerator.setBounds(new Rectangle(7, 23, 70, 184));
 Accelerator.addMouseListener(new java.awt.event.MouseAdapter() {

  public void mouseReleased(MouseEvent e) {
   Accelerator_mouseReleased(e);
  }
 });
 Accelerator.addChangeListener(new javax.swing.event.ChangeListener() {

  public void stateChanged(ChangeEvent e) {
   Accelerator_stateChanged(e);
  }
 });
 RPM.setOrientation(JProgressBar.VERTICAL);
 RPM.setMaximum(7500);
 RPM.setStringPainted(true);
 RPM.setBounds(new Rectangle(10, 8, 50, 164));
 StartStop.setText("Start The Vehicle");
 StartStop.setBounds(new Rectangle(9, 232, 195, 30));
 StartStop.addMouseListener(new java.awt.event.MouseAdapter() {

  public void mouseClicked(MouseEvent e) {
   StartStop_mouseClicked(e);
  }
 });
 jPanel1.setBorder(BorderFactory.createRaisedBevelBorder());
 jPanel1.setBounds(new Rectangle(141, 25, 70, 181));
 jPanel1.setLayout(null);
 this.setResizable(false);
 this.setTitle("Actual Vehicle");
 this.getContentPane().add(jLabel1, null);
 this.getContentPane().add(Accelerator, null);
 this.getContentPane().add(jLabel2, null);
 this.getContentPane().add(jLabel3, null);
 this.getContentPane().add(jLabel4, null);
 this.getContentPane().add(jLabel5, null);
 this.getContentPane().add(LabelOnOff, null);
 this.getContentPane().add(StartStop, null);
 this.getContentPane().add(jPanel1, null);
 jPanel1.add(RPM, null);
}
```

```
 void set_enable(boolean en){
  System.out.println("Got to CGUI, enabling...");
  enabled = en;

  if(!enabled)
  {   this.setTitle("Copy of Actual Vehicle");
      this.Accelerator.setEnabled(false);
      this.StartStop.setEnabled(false);
  }
 }

 void StartStop_mouseClicked(MouseEvent e) {
   if(enabled){
      if(LabelOnOff.getText() == "OFF")
      { LabelOnOff.setText("ON");
        StartStop.setText("Stop The Vehicle");
        vehicle.set_running(true);
      }
      else
      {  LabelOnOff.setText("OFF");
         StartStop.setText("Start The Vehicle");
         vehicle.set_running(false);
      }
   }
   else {
      Accelerator.setEnabled(false);
      StartStop.setEnabled(false);
   }
 }

 void Accelerator_stateChanged(ChangeEvent e) {
   if(enabled)
      vehicle.set_acc((short)Accelerator.getValue());
   else
      Accelerator.setEnabled(false);

 }
void Accelerator_mouseReleased(MouseEvent e) {
   if(enabled)
      Accelerator.setValue(0);
   else
      Accelerator.setEnabled(false);
 }
}
```

cietm.java:

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;
import javax.swing.UIManager;
import javax.swing.UIManager;
```

```
import org.omg.PortableServer.*;
import org.omg.CosNaming.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.beans.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import java.beans.*;
import javax.swing.UIManager;

public class Cietm {
  static VehicleModule.vehicleServer vehicle;
  static String  [] args;
  static String cname = "IETMClient";
  static int index = 0;
  static boolean en;

//Construct the application
  Cietm(String [] a) {
     args = a;
     try{
      //Initialize the ORB
      System.out.println("Starting CORBA for client client");
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

      System.out.println("CC ORB Initialized");
      //obtain the root context
      org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
      System.out.println("CC Naming converted...");
      NamingContext root = NamingContextHelper.narrow(rootObj);
      System.out.println("CC Nameing narrowed...");
      //Locate an account manager through the naming service
      NameComponent[] qs_name = new NameComponent[1];
      qs_name[0] = new NameComponent("Vehicle","");
      org.omg.CORBA.Object vehicleObj = root.resolve(qs_name);
      vehicle = VehicleModule.vehicleServerHelper.narrow(vehicleObj);
      index = vehicle.get_index();
      System.out.println("Client #" + index + " is ready");
      cname += index;
      }
      catch (Exception e){
        e.printStackTrace();
      }

   //Start up CORBA ORB
    try{
      //Initialize the orb
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args, null);
      System.out.println("Client Server ORB initialized");
```

```
    //Get a reference to the root POA
    POA rootPOA = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

    //Get a reference to the Naming Service root_context
    org.omg.CORBA.Object rootObj = orb.string_to_object(args[0]);
    NamingContext root = NamingContextHelper.narrow(rootObj);
    System.out.println("CS Naming Narrowed...");
    //Create policies for our Persistant POA
    org.omg.CORBA.Policy[] policies = {
      rootPOA.create_lifespan_policy(LifespanPolicyValue.PERSISTENT),

rootPOA.create_request_processing_policy(RequestProcessingPolicyValue.USE_DEFAULT_SERVANT)
    };

    //Create the POA with above policies
    POA myPOA = rootPOA.create_POA(cname+"_POA",
    rootPOA.the_POAManager(), policies);
    System.out.println("CS POA Created ...");
    //Create the Servant
    IVehicle myServant = new IVehicle(vehicle);
    myPOA.set_servant(myServant);
    org.omg.CORBA.Object ref;
    System.out.println("CC Servant initialized...");
    //Decide on the ID for the servant
    byte[] myservantId = cname.getBytes();
    System.out.println("CC Servant ID gotten...");
    //Activate the POA Manager
    myPOA.activate_object_with_id(myservantId, myServant);
    ref = myPOA.create_reference_with_id(cname.getBytes(),
        "IDL:VehicleModule/" + cname + ":1.0");
    System.out.println("CC Activate object...");
    //Activate the POA Manager
    rootPOA.the_POAManager().activate();
    System.out.println("CC POA Manger activate...");
    NameComponent [] qs_name = new NameComponent[1];
    qs_name[0] = new NameComponent(cname,"");
    System.out.println("CC Name Component initialized...");

    ((NamingContext)root).bind(qs_name,myPOA.servant_to_reference(myServant));
    System.out.println("CC Context bound.... Running ORB");

    System.out.println("CC registering client...");
    en = vehicle.add_client(orb.object_to_string(ref),cname);
   //Enabling Client doesnt work
    System.out.println("CC calling enabled... Enabled = " + en);
    myServant.set_enable(en);
    System.out.println("CC enabled set...");


    System.out.println("CC Running ORB");
    orb.run();

  }
  catch (Exception e){
    e.printStackTrace();
  }
```

```
  }

  //Main method
  public static void main(String[] a) {
    System.out.println("Starting IETM Client Main");
    try {
      UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    new Cietm(a);
  }
}

class IVehicle extends VehicleModule.vehicleClientPOA
{
  static VehicleModule.vehicleServer vehicle;
  static IGUI frame;
  boolean enabled = true;

  IVehicle(VehicleModule.vehicleServer v){
    vehicle = v;
    frame = new IGUI(vehicle);
    frame.setVisible(true);
  }

  public void set_enable(boolean en){
    System.out.println("got to CVehicle, enabling...En = " + en);
    enabled = en;
    System.out.println("Going to CGUI...Enabled = " + enabled);
    frame.set_enable(true);
    System.out.println("Going to CGUI...Enabled = " + enabled);
    frame.set_enable(enabled);
  }

public void set_rpm(short rpm){
    frame.jLabel4.setText("RPM = " + rpm);
  }

  public void set_testparm(String parm){
    frame.ADecelerate.setText(parm);
  }

  public void results(float accel, float pcf){
    if(pcf >=1 && pcf <=100)
      frame.ADecelerate.setText((int)pcf + "%");
    else
      frame.ADecelerate.setText("");

    int tens = (int)(accel/1000);
    int ones = (int)(accel - 1000*((int)(accel/1000)));
    if(ones == 0)
      frame.jLabel4.setText(tens + ",000 RPM/s");
    else
      frame.jLabel4.setText(tens + "," + ones + " RPM/s");
```

```
    frame.testrunningtext.setForeground(Color.black);
    frame.testrunningtext.setText("Test Completed");
    frame.stopbutton.setText("Return to Menu");
  }
}

class IGUI extends JFrame{
 static int vid = 123433;
 static int tstnum = 0;
 public boolean power = false;
 static VehicleModule.vehicleServer vehicle;
 JLabel ADecelerate = new JLabel();
 static boolean enabled;
 JPanel jPanel1 = new JPanel();
 JLabel jLabel4 = new JLabel();
 JLabel runatest = new JLabel();
 JLabel Test12 = new JLabel();
 JTextPane jTextPane1 = new JTextPane();
 JTextPane testrunningtext1 = new JTextPane();
 JLabel Test13 = new JLabel();
 JLabel Title = new JLabel();
 JButton runbutton = new JButton();
 JPanel jPanelRPMs = new JPanel();
 JPanel jPanelRun = new JPanel();
 JPanel jPanelchoose = new JPanel();
 JPanel jPanelgo = new JPanel();
 JButton stopbutton = new JButton();
 JPanel jPanelstop = new JPanel();
 JLabel testrunningtext = new JLabel();
 JProgressBar RPM = new JProgressBar();
 JSlider Accelerator = new JSlider();


 IGUI(VehicleModule.vehicleServer v) {
  vehicle = v;

  //test_power t = new test_power(this, vehicle);
  //Thread tp = new Thread(t);
  //tp.start();

  try {
   jbInit();
  }
  catch(Exception e) {
   e.printStackTrace();
  }

 }

 private void jbInit() throws Exception {
   this.jPanelRun.setVisible(true);
   this.jPanelRun.setEnabled(true);
   this.jPanelstop.setVisible(false);
   this.jPanelstop.setEnabled(false);
   this.jPanelRPMs.setVisible(false);
   this.jPanelgo.setVisible(false);
```

```
 this.jPanelchoose.setVisible(false);
 this.runbutton.setEnabled(false);
 this.runbutton.setVisible(false);
 this.stopbutton.setEnabled(false);
 this.stopbutton.setVisible(false);
 this.ADecelerate.setText("");

this.setSize(275,360);
this.getContentPane().setLayout(null);
ADecelerate.setFont(new java.awt.Font("SansSerif", 1, 26));
ADecelerate.setForeground(Color.red);
ADecelerate.setBorder(BorderFactory.createEtchedBorder());
ADecelerate.setHorizontalAlignment(SwingConstants.CENTER);
ADecelerate.setHorizontalTextPosition(SwingConstants.CENTER);
ADecelerate.setText("Accelerate");
ADecelerate.setBounds(new Rectangle(6, 52, 187, 34));
Accelerator.setValue(0);
Accelerator.setOrientation(JSlider.VERTICAL);
Accelerator.setSnapToTicks(true);
Accelerator.setMajorTickSpacing(25);
Accelerator.setPaintTicks(true);
Accelerator.setMinorTickSpacing(5);
Accelerator.setPaintLabels(true);
Accelerator.setToolTipText("");
Accelerator.setBorder(BorderFactory.createRaisedBevelBorder());
Accelerator.setBounds(new Rectangle(7, 23, 70, 184));
this.setResizable(false);
this.setTitle("IETM");
jPanel1.setBackground(Color.white);
jPanel1.setBorder(BorderFactory.createLoweredBevelBorder());
jPanel1.setBounds(new Rectangle(11, 7, 198, 89));
jPanel1.setLayout(null);
jLabel4.setText("RPM =");
jLabel4.setBounds(new Rectangle(5, 5, 187, 34));
jLabel4.setForeground(Color.black);
jLabel4.setToolTipText("");
jLabel4.setFont(new java.awt.Font("SansSerif", 1, 26));

runatest.setForeground(Color.blue);
runatest.setText("Run A Power Test");
runatest.setBounds(new Rectangle(8, 9, 203, 17));
runatest.addMouseListener(new java.awt.event.MouseAdapter() {

 public void mouseClicked(MouseEvent e) {
  runatest_mouseClicked(e);
 }
});
Test12.setBounds(new Rectangle(5, 7, 182, 31));
Test12.addMouseListener(new java.awt.event.MouseAdapter() {

 public void mouseClicked(MouseEvent e) {
  Test12_mouseClicked(e);
 }
});
Test12.setForeground(Color.blue);
Test12.setDisplayedMnemonic('0');
```

```
Test12.setHorizontalAlignment(SwingConstants.LEFT);
Test12.setHorizontalTextPosition(SwingConstants.LEFT);
Test12.setText("Test 12: TK Power Test");
jTextPane1.setBackground(Color.lightGray);
jTextPane1.setBorder(BorderFactory.createRaisedBevelBorder());
jTextPane1.setText("Click on a highligted item below to choose what you want to
do...");
jTextPane1.setFont(new java.awt.Font("SansSerif", 1, 14));
jTextPane1.setBounds(new Rectangle(10, 56, 252, 52));
jTextPane1.setEnabled(false);
Test13.setText("Test 13: DCA Power Test");
Test13.setHorizontalTextPosition(SwingConstants.LEFT);
Test13.setHorizontalAlignment(SwingConstants.LEFT);
Test13.setDisplayedMnemonic('0');
Test13.setBounds(new Rectangle(7, 36, 179, 31));
Test13.addMouseListener(new java.awt.event.MouseAdapter() {

 public void mouseClicked(MouseEvent e) {
  Test13_mouseClicked(e);
 }
});
Test13.setForeground(Color.blue);
Title.setFont(new java.awt.Font("SansSerif", 1, 36));
Title.setForeground(Color.black);
Title.setBorder(BorderFactory.createRaisedBevelBorder());
Title.setHorizontalAlignment(SwingConstants.CENTER);
Title.setText("IETM DEMO");
Title.setBounds(new Rectangle(10, 6, 252, 47));
runbutton.setToolTipText("");
runbutton.setEnabled(false);
runbutton.setVisible(false);
runbutton.setText("Run Test");
runbutton.setBounds(new Rectangle(9, 56, 204, 41));
runbutton.addMouseListener(new java.awt.event.MouseAdapter() {

 public void mouseClicked(MouseEvent e) {
  runbutton_mouseClicked(e);
 }
});
jPanelRPMs.setAlignmentX((float) 0.2);
jPanelRPMs.setAlignmentY((float) 0.2);
jPanelRPMs.setBorder(BorderFactory.createRaisedBevelBorder());
jPanelRPMs.setBounds(new Rectangle(26, 221, 220, 105));
jPanelRPMs.setLayout(null);
jPanelRun.setBorder(BorderFactory.createLoweredBevelBorder());
jPanelRun.setBounds(new Rectangle(26, 111, 220, 105));
jPanelRun.setLayout(null);
jPanelRun.setEnabled(true);
jPanelchoose.setLayout(null);
jPanelchoose.setVisible(false);
jPanelchoose.setBounds(new Rectangle(26, 111, 220, 105));
jPanelchoose.setBorder(BorderFactory.createLoweredBevelBorder());
jPanelgo.setBorder(BorderFactory.createLoweredBevelBorder());
jPanelgo.setBounds(new Rectangle(26, 111, 220, 105));
jPanelgo.setLayout(null);
jPanelgo.setEnabled(false);
```

```
jPanelgo.setVisible(false);
stopbutton.setBounds(new Rectangle(12, 56, 197, 42));
stopbutton.addMouseListener(new java.awt.event.MouseAdapter() {

  public void mouseClicked(MouseEvent e) {
    stopbutton_mouseClicked(e);
  }
});
stopbutton.setText("Stop Test");
stopbutton.setEnabled(false);
stopbutton.setVisible(false);
stopbutton.setToolTipText("");
stopbutton.setActionCommand("Stop Test");
jPanelstop.setLayout(null);
jPanelstop.setBounds(new Rectangle(26, 111, 220, 105));
jPanelstop.setEnabled(false);
jPanelstop.setBorder(BorderFactory.createLoweredBevelBorder());
testrunningtext.setForeground(Color.red);
testrunningtext.setBounds(new Rectangle(11, 8, 179, 31));
testrunningtext.setDisplayedMnemonic('0');
testrunningtext.setHorizontalAlignment(SwingConstants.LEFT);
testrunningtext.setHorizontalTextPosition(SwingConstants.LEFT);
testrunningtext.setText("Running Power Test...");
testrunningtext1.setEnabled(false);
testrunningtext1.setText("Please be sure vehicle is on and idling.");
testrunningtext1.setBounds(new Rectangle(9, 10, 179, 65));
testrunningtext1.setForeground(Color.black);
testrunningtext1.setBackground(Color.lightGray);
RPM.setEnabled(false);
RPM.setBounds(new Rectangle(412, 50, 57, 27));
RPM.setVisible(false);
Accelerator.setEnabled(false);
Accelerator.setBounds(new Rectangle(475, 92, 62, 32));
Accelerator.setVisible(false);
this.getContentPane().add(Title, null);
this.getContentPane().add(RPM, null);
this.getContentPane().add(Accelerator, null);
this.getContentPane().add(jTextPane1, null);
this.getContentPane().add(jPanelRun, null);
jPanelRun.add(runatest, null);
this.getContentPane().add(jPanelchoose, null);
jPanelchoose.add(Test13, null);
jPanelchoose.add(Test12, null);
this.getContentPane().add(jPanelstop, null);
jPanelstop.add(stopbutton, null);
jPanelstop.add(testrunningtext, null);
this.getContentPane().add(jPanelgo, null);
jPanelgo.add(runbutton, null);
jPanelgo.add(testrunningtext1, null);
this.getContentPane().add(jPanelRPMs, null);
jPanelRPMs.add(jPanel1, null);
jPanel1.add(jLabel4, null);
jPanel1.add(ADecelerate, null);
runatest.setEnabled(true);
runatest.setVisible(true);
this.jPanelRun.setEnabled(true);
```

```
    this.jPanelRun.setVisible(true);
  }

  void set_enable(boolean en){
    System.out.println("Got to CGUI, enabling...");
    enabled = en;

    if(!enabled)
    {   this.setTitle("Copy of IETM");
        this.Accelerator.setEnabled(false);
    }
  }



  void stopbutton_mouseClicked(MouseEvent e) {
    System.out.println("Starting stop button");
    this.jPanelstop.setVisible(false);
    this.jPanelstop.setEnabled(false);
    this.jPanelRun.setVisible(true);
    this.jPanelRun.setEnabled(true);
    this.jPanelRPMs.setVisible(false);
    this.jPanelRPMs.setEnabled(false);
    this.runbutton.setEnabled(false);
    this.runbutton.setVisible(false);
    this.testrunningtext1.setVisible(false);
    this.testrunningtext1.setEnabled(false);
  }

  void runbutton_mouseClicked(MouseEvent e) {
    System.out.println("Run Clicked");
    testrunningtext.setForeground(Color.red);
    stopbutton.setText("Stop Test");
    testrunningtext.setText("Running Power Test...");
    this.ADecelerate.setText("");

    this.runbutton.setEnabled(false);
    this.runbutton.setVisible(false);
    this.testrunningtext1.setVisible(false);
    this.jPanelgo.setVisible(false);
    this.jPanelgo.setEnabled(false);
    this.jPanelRPMs.setVisible(true);
    this.jPanelRPMs.setEnabled(true);
    this.jPanelstop.setVisible(true);
    this.jPanelstop.setEnabled(true);
    this.stopbutton.setEnabled(true);
    this.stopbutton.setVisible(true);
    System.out.println("Panels Changed, starting test_power");

    test_power t = new test_power(this, vehicle);
    Thread tp = new Thread(t);
    tp.start();
  }
```

```
    void Test13_mouseClicked(MouseEvent e) {
      System.out.println("Starting Test13");
      tstnum = 12;
      ADecelerate.setText("");
      this.jPanelchoose.setVisible(false);
      this.jPanelchoose.setEnabled(false);
      this.jPanelgo.setVisible(true);
      this.jPanelRPMs.setVisible(true);
      this.jPanelRPMs.setEnabled(true);
      this.runbutton.setEnabled(true);
      this.runbutton.setVisible(true);
      this.testrunningtext1.setVisible(true);
      this.jLabel4.setText("RPM = ");
    }

    void Test12_mouseClicked(MouseEvent e) {
      System.out.println("Starting Test12");
      tstnum = 12;
      ADecelerate.setText("");
      this.jPanelchoose.setVisible(false);
      this.jPanelchoose.setEnabled(false);
      this.jPanelgo.setVisible(true);
      this.jPanelRPMs.setVisible(true);
      this.jPanelRPMs.setEnabled(true);
      this.runbutton.setEnabled(true);
      this.runbutton.setVisible(true);
      this.testrunningtext1.setVisible(true);
      this.jLabel4.setText("RPM = ");
    }
  void runatest_mouseClicked(MouseEvent e) {
      System.out.println("Starting runatest");
      this.jPanelRun.setVisible(false);
      this.jPanelRun.setEnabled(false);
      this.jPanelchoose.setVisible(true);
      this.jPanelchoose.setEnabled(true);
      this.jPanelRPMs.setVisible(false);
      this.jPanelRPMs.setEnabled(false);
    }
  }
  class test_power implements Runnable{
    IGUI frame;
    VehicleModule.vehicleServer vehicle;

    test_power(IGUI f, VehicleModule.vehicleServer v){
      frame = f;
      vehicle = v;
    }


    public void run(){
      int status = 0;
      int i;
  //    frame.jPanelstop.setVisible(true);
  //    frame.jPanelstop.setEnabled(true);

      for(i = 0; i < 6; ++i)
```

```
    {   if(i == 3) ++i;
        frame.testrunningtext.setText("Running Power Test..." + i);
        if(status == 0)
          status = vehicle.test_power((short)i);
        else break;
    }
    if(status != 0)
    {   System.out.println("IETM Test Power: Status returned was not 0");
        frame.jLabel4.setText("Error " + i + ":" + status);
        frame.testrunningtext.setText("Error " + i + ":" + status + "." );
        frame.ADecelerate.setText("");
    }
  }
}
```

Appendix K:

CORBA Power Test Demonstration Software, C++ Server, for Windows NT

vehiclemodule.idl:

```
module VehicleModule
{  interface vehicleServer
        {       long test_power(in short step);
                void set_running(in boolean running);
           void set_acc(in short acc);
                boolean set_baseline(in float acc);
                short get_index();
                boolean add_client(in string ior, in string name);
        };

        interface vehicleClient
        {       void set_rpm(in short rpm);
                void set_testparm(in string parm);
                void results(in float accel, in float pcf);
        };
};
```

sserver.cpp:

```
/*************************************************************************/
/* File             SServer.cpp
                                                                */
/* Class  SServer
                                                        */
/*************************************************************************/

#include <stdio.h>
#include "powertypes.h"
#include "m:\TAO_ORB\TAO_zip_version\ACE_wrappers\TAO\orbsvcs\orbsvcs\CosNamingC.h"
#include <iostream>
#include <sys/types.h>
#include <sys/timeb.h>

/*************************************************************************/
/* Function                 main                                                   */
/* Description     Starts the server                                   */
/* Date                     8/16/2000                                          */
/* Modification                                                        */
/* References                                                                  */
/*-------------------------------------------------------------------------*/
/* PreCond                  none                                               */
/*-------------------------------------------------------------------------*/
/* PostCond                 SServer will be started                    */
/*************************************************************************/
int main(int argc, char *argv[])
{       std::cout << "argc = " << argc << " argv[0] = " << argv[0] << std::endl;
        SServer *s = new SServer(argc, argv);
        return 0;
}/* End of function main
                                                        */
```

```
/*************************************************************************/
/* Class          SServer                                                */
/* Function       SServer                                                */
/* Description    Constructor creates server and initializes value       */
/* Date           8/16/2000                                              */
/* Modification                                                          */
/* References                                                            */
/*-----------------------------------------------------------------------*/
/* PreCond        none                                                   */
/*-----------------------------------------------------------------------*/
/* PostCond       SServer will be started and initialized                */
/*************************************************************************/
SServer::SServer(int argc, char *argv[])
{       //Start up CORBA ORB
        try
        {       std::cout << "Initializing ORB...\n" << std::endl;
                CORBA::ORB_var orb = CORBA::ORB_init (argc, argv,"");
          CORBA::Object_var poa_object = orb->resolve_initial_references
            ("RootPOA");

                std::cout << "Initializing POA...\n" << std::endl;
                 PortableServer::POA_var poa = PortableServer::POA::_narrow    (poa_object.in ());
                PortableServer::POAManager_var poa_manager = poa-
             >the_POAManager ();
                poa_manager->activate();

                // Create the servant
                std::cout << "Creating Servant...\n" << std::endl;
                SPowerTest * servant = new SPowerTest();

                // Activate it to obtain the object reference
                std::cout << "Activating Servant...\n" << std::endl;
                VehicleModule::vehicleServer_var myobject=(*servant)._this();

                // Get the Naming Context reference
                std::cout << "Getting Naming Service...\n" << std::endl;
                CORBA::Object_var naming_context_object = orb-
              >resolve_initial_references ("NameService");
                CosNaming::NamingContext_var naming_context =
             CosNaming::NamingContext::_narrow (naming_context_object.in ());

                // Create and initialize the name.
                CosNaming::Name name (1);
                name.length (1);
                name[0].id = CORBA::string_dup ("Vehicle");

                // Bind the object
                std::cout << "Naming Context Bound...\n" << std::endl;
                naming_context->bind (name, myobject.in ());

                std::cout << "Server is running... \nStart clients now..." << std::endl;
                orb->run ();
```

```
                // Destroy the POA, waiting until the destruction terminates
                poa->destroy (1, 1);
                orb->destroy ();
        ;}
        catch (CORBA::Exception &)
        {       std::cerr << "CORBA exception raised!" << std::endl;
        }
}/* End of constructor SServer::SServer*/
```

vehicleframe.cpp:

```
/*************************************************************************/
/* File             VehicleFrame.cpp
                                                        */
/* Class  VehicleFrame
                                                */
/*************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "powertypes.h"
#include <sys/types.h>
#include <sys/timeb.h>
#include <string.h>

/*************************************************************************/
/* Class            VehicleFrame
                                                */
/* Function                 VehicleFrame
                                                        */
/* Description      Constructs the vehicle frame, initializing all values                   */
/* Date                     8/16/2000
                                                                */
/* Modification
                                                                */
/* References
                                                                */
/*-------------------------------------------------------------------------*/
/* PreCond                  none
                                                                */
/*-------------------------------------------------------------------------*/
/* PostCond                 All internal values will be initialized
                                */
/*************************************************************************/
VehicleFrame::VehicleFrame()
{       setvid(123433);
        setRPM(0);
        setAccelerator(0);
        setStartStop(false);
        setLabelOnOff("Off");
        setjLabel4(0);
}/* End of VehicleFrame Constructor
                                                */


/*************************************************************************/
```

```
/* Class          VehicleFrame
                                                    */
/* Function               VehicleFrame(vehicle ID)
                                                    */
/* Description    Constructs the vehicle frame, initializing all values and setting */
/*                                                  vehicle ID to input parameter
                                                    */
/* Date                   8/16/2000
                                                                    */
/* Modification
                                                                    */
/* References     VehicleFrame()
                                                    */
/*-------------------------------------------------------------------------------*/
/* PreCond                v is a valid vehicle ID
                                                    */
/*-------------------------------------------------------------------------------*/
/* PostCond               vehicle ID will be set to input parameter
                */
/*                                  All internal values will be initialized
                                                    */
/*********************************************************************/
VehicleFrame::VehicleFrame(int v)
{       setvid(v);
        setRPM(0);
        setAccelerator(0);
        setStartStop(false);
        setLabelOnOff("Off");
        setjLabel4(0);
}/* End of VehicleFrame Constructor w/VID                                  */

/*********************************************************************/
/* Class          VehicleFrame
                                                    */
/* Function               Set Functions
                                                            */
/* Description    Set the individual values within the VehicleFrame
        */
/* Date                   8/16/2000
                                                                    */
/* Modification
                                                                    */
/* References
                                                                    */
/*-------------------------------------------------------------------------------*/
/* PreCond                none
                                                                    */
/*-------------------------------------------------------------------------------*/
/* PostCond               Specific value will be set                              */
/*********************************************************************/
void VehicleFrame::setAccelerator(int acc)
{       Accelerator = acc;
}/* End of function setAccelerator from Class VehicleFrame                 */

void VehicleFrame::setjLabel4(int rpm)
```

```
{       char temp[11];    /* Temporarily holds the new label text
        */

        strcpy(temp, "RPM = ");
        _itoa(rpm, temp,10);
        strcpy(jLabel4, temp);
}/* End of function setjLabel4 from Class VehicleFrame                    */

void VehicleFrame::setLabelOnOff(char onoff[])
{       strcpy(LabelOnOff, onoff);
}/* End of function setjLabel4 from Class VehicleFrame                    */

void VehicleFrame::setRPM(int rpm)
{       RPM = rpm;
}/* End of function setRPM from classs VehicleFrame                       */

void VehicleFrame::setStartStop(bool running)
{       StartStop = running;
}/* End of funciton setStartStop form class VehicleFrame                  */

void VehicleFrame::setvid(int v)
{       vid = v;
}/* End of function setvid from class VehicleFrame                        */

/*************************************************************************/
/* Class          VehicleFrame                                           */
/* Function               Get Functions                                          */
/* Description    Get the individual values within the VehicleFrame & return to caller*/
/* Date                   8/16/2000                                              */
/* Modification                                                          */
/* References                                                                    */
/*-------------------------------------------------------------------------*/
/* PreCond                none                                                   */
/*-------------------------------------------------------------------------*/
/* PostCond               Specific value will be returned
                                */
/*************************************************************************/
int VehicleFrame::getRPM()
{       return RPM;
}/* End of function getRPM from class VehicleFrame
                        */

int VehicleFrame::getvid()
{       return vid;
}/* End of function getvid from class VehicleFrame
                */

int VehicleFrame::getAccelerator()
{       return Accelerator;
}/* End of function getvid from class VehicleFrame                        */

bool VehicleFrame::getStartStop()
{       return StartStop;
}/* End of function getvid from class VehicleFrame
```

cnode.cpp:

```
/*************************************************************************/
/* File              cnode.cpp
                                                                         */
/* Class  cnode
                                                                      */
/*************************************************************************/

#include <stdio.h>
#include <string.h>
#include "vehiclemoduleC.h"
#include "powertypes.h"

/*************************************************************************/
/* Class             cnode
                                                                      */
/* Function                  cnode
                                                                      */
/* Description     cnode Constructor initializes new cnode to null
          */
/* Date                      8/16/2000
                                                                      */
/* Modification
                                                                      */
/* References
                                                                      */
/*-----------------------------------------------------------------------*/
/* PreCond                   none
                                                                      */
/*-----------------------------------------------------------------------*/
/* PostCond                  New cnode will be empty
                                                                   */
/*************************************************************************/
cnode::cnode()
{         /* Initialize components to null
                                                                   */

        vclient = NULL;
   next = NULL;
   strcpy(name,"");

}/* End of cnode constructor                                                */
```

spowertest.cpp:

```
/*************************************************************************/
/* File              SPowerTest.cpp
                                                                      */
/* Class  SPowerTest
                                                                      */
/*************************************************************************/

#include <stdio.h>
#include "powertypes.h"
#include "m:\TAO_ORB\TAO_zip_version\ACE_wrappers\TAO\orbsvcs\orbsvcs\CosNamingC.h"
#include <iostream>
#include <sys/types.h>
#include <sys/timeb.h>
```

```
/*************************************************************************/
/* Class            SPowerTest
                                                                    */
/* Function
                                                                       */
/* Description
                                                                    */
/* Date                    8/16/2000
                                                                    */
/* Modification
                                                                    */
/* References
                                                                    */
/*-----------------------------------------------------------------------*/
/* PreCond
                                                                       */
/*-----------------------------------------------------------------------*/
/* PostCond
                                                                       */
/*************************************************************************/
SPowerTest::SPowerTest()
{   frame = new VehicleFrame;
}/* End of SPowerTest Class Constructor
                                    */


/*************************************************************************/
/* Class            SPowerTest
                                                                    */
/* Function
                                                                       */
/* Description
                                                                    */
/* Date                    8/16/2000
                                                                    */
/* Modification
                                                                    */
/* References
                                                                    */
/*-----------------------------------------------------------------------*/
/* PreCond
                                                                       */
/*-----------------------------------------------------------------------*/
/* PostCond
                                                                       */
/*************************************************************************/
int SPowerTest::GetThreshRPM(int thresh, struct _timeb *time, int step, float *rpm)
{           struct _timeb currtime,  /* The current system time
                    */
                                    timeout;   /* The time at which a timeout occurs
                    */
```

```
        /* Check to see if in accel or decel portion of test
        */
        if(step == 2)
        {       /* In deceleration portion of test timeout = 3 seconds
        */
                //std::cout << "ThreshRPM step 2 started..." << std::endl;
                timeout.time = (*time).time + 3;
                //std::cout << "ThreshRPM step 2 timeout = " << timeout.time <<
               std::endl;
                update_clients("Decelerate");
        }
        else
        {       /* In Acceleration portion of test
                                */
                update_clients("Accelerate");
                if(step == 0)
                {       std::cout << "ThreshRPM step 0 started..." << std::endl;
                        /* Start of accel portion of test, timout = 20 seconds
        */
                        timeout.time = (*time).time + 20;
                        //std::cout << "ThreshRPM Timeout+20="<< timeout.time <<
                     std::endl;
                        //std::cout << "ThreshRPM time->time = "<< time->time <<
                      std::endl;
                        //std::cout << "ThreshRPM time = "<< time << std::endl;
                }
                else
                {       //std::cout << "ThreshRPM step 1 started..." << std::endl;
                        /* Second accel portion of test, timout = 2 seconds
                */
                        timeout.time = (*time).time + 2;
                }
        }/* End if
                                                                        */
//check for timeout seconds for rpm to exceed thresh
  do
  {  //std::cout << "ThreshRPM Getting curr time..." << std::endl;
                /* Get the current time and rpm
                                        */
                _ftime(&currtime);
                //std::cout << "ThreshRPM curr time = "<< currtime.time << std::endl;
                *rpm = frame->getRPM();
                //std::cout << "ThreshRPM RPM = " << frame->getRPM() << std::endl;

                /* Check to see if rpm has met thresholds
                        */
                if((step != 2  && *rpm > thresh) || (step == 2 && *rpm < thresh))
    {   //std::cout << "ThreshRPM RPM has met Threshold (" << thresh << ":"<<*rpm
            << std::endl;
                        /* RPM has met threshold, set time and return ok
                */
                        //std::cout << "ThreshRPM returning curr time" << std::endl;
                        *time = currtime;
                        //std::cout << "ThreshRPM curr time="<<currtime.time <<
                     std::endl;
        return 0;
```

```
                }
    }while(currtime.time  < timeout.time);

        //std::cout << "ThreshRPM Returning Error" << std::endl;
        //std::cout << "ThreshRPM curr time="<<currtime.time<< "
        timeout="<<timeout.time << std::endl;
        /* Timeout has occured so return BAD_DATA error
                        */
    return BAD_DATA;
}/* End of function GetThreshRPM of class SPowerTest
                */

/*************************************************************************/
/* Class            SPowerTest
                                                */
/* Function                 test_power
                                                        */
/* Description      Measures CI engine power.
                        */
/*                                  Test 12 uses the DCA tachometer.
                        */
/*                                  Test 13 uses the TK tachometer.
                                    */
/* Date
                                                                */
/* Modification     8/16/2000
                                                */
/* References       US Army PM-TMDE test_power
                        */
/*-------------------------------------------------------------------------*/
/* PreCond                  If step = 0, Do nothing
                                    */
/*                                          1, Set up hardware & parameters for the test.
                        */
/*                                          3, Compute the latest test result while the test is
        */
/*                                                  looping during the idle/governor speed
check,                  */
/*                                          4, Run the acceleration portion of the test.
                */
/*                                          5, Run the deceleration portion of the test & finish
up.     */
/*-------------------------------------------------------------------------*/
/* PostCond                 Returns:  0 -> test OK
                                    */
/*                                          -n -> error n
                                                        */
/*************************************************************************/
/*   Notes:
                                                        */
/*   Removed lines that called outside functions not available to the demo                  */
/*     (i.e. relay setpoints).
                        */
/*   Removed lines that used global variables not pertinent to demo (i.e. DATA) */
/*   Changed function GetTickCount to system time function
        */
```

```
/*   moved static variables into class description
                    */
/*   changed ovcnt from WORD to int
                                        */
/*   Changed Value to Current RPM
                                                */
/*          Added rpm2
                                                                */
/*************************************************************************/
int SPowerTest::test_power(short step) /* Step that the test is on
          */
{         int ret;  /* The return value if not 0
                              */
/*        float *Data[2];
                                                                */
// The following added in place of future corba functions
          int tstnum[] = {12,0};
  char Units[4] = {""};
// Use Data to store past results
/*        Data[0] = Value[0];
                                                                */
/*        Data[1] = Value[1];
                                                                */

          std::cout << "Test_Power recieved " << step << std::endl;
          // Comm check, no work done:
          if(step == 0) return 0;

          // Set up for the idle/governor check and/or main test:
          else if(step == 1)
          {         // Demo: Initialize all vars
    rpm1 = 0;                 // Holds first RPM during test
    rpm2 = 0;       // Added to hold second RPM value
    pcf = 0;        // Added to hold percentage value
    thresh1 = 0;    // Lower RPM Threshold
    thresh2 = 0;    // Upper RPM Threshold
    accel = 0;      // Acceleration value
    ovcnt = 0;                // Counts the number of glitches in Gov

                    if(tstnum[0] == 12)
                    {         // Next 2 lines removed for demo
                              /* if( (ret = pRelay(c_d, 0, CT1F)) < 0) return ret;
        */
          /* r_p = rev_pulse;
                              */
                    }
                    else
                    {         // Next 2 lines removed for demo
                              /* if( (ret = pRelay(TKrelay[0], 0, CT1F)) < 0) return ret;
        */
                                        /* r_p = 1.;
                                                                */
                    }

                    // Next 2 lines removed for demo
```

```
            /* if( (ret = pCounter(0, 1, CG_1M)) < 0) return ret;
    */
            /* Mult[0] = r_p * 6.0e4;
                                        */

      strcpy(Units,"rpm");
      std::cout << "Finished Power_Test(1)" << std::endl;
      return 0;
    }

    // Compute idle, governor check running test value:
    else if(step == 3)
    {       ret = 0; /* Originally set to InterValue(0);
            */

// Deglitch spikes which get thru tach filter:
      if(ret < 0)
            //Deglitch
  if(++ovcnt < 4) ret = 0;
      else ovcnt = 0;

            std::cout << "Finished Power_Test(3)" << std::endl;
      return ret;
    }

    // Run the acceleration portion of the test:
    else if(step == 4)
    {  // Prepare for Power test:
            if(frame->getvid() == 21 || frame->getvid() == 23)
      {  thresh1 = 2000;
                  thresh2 = 3000;
      }
      else
      {  thresh1 = 1000;
                  thresh2 = 2000;
      }

            // 20 sec to get started
            _ftime(&t2);
            //std::cout << "TestPower t2 = "<< t2.time << std::endl;
      t2.time += 20;
            //std::cout << "TestPower t2+20 = "<< t2.time << std::endl;

// Make sure engine is still running and get start time & rpm:
            while(frame->getRPM() <= 0)
      {     //Following removed for demo
  /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;                    */
  update_clients("Vehicle Is Off!");
  _ftime(&t1);
                  //std::cout << "TestPower t1 = "<< t1.time << std::endl;
                  if( t1.time > t2.time) return TIMEOUT;
      }

      rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];

// Must start with the rpm < thresh1:
```

```
        /* if( (Value[0][0] = rpm1) > thresh1) return INDETERM;
      */
if( rpm1 > thresh1) return INDETERM;

                _ftime(&t1);
// Loop until rpm > thresh1 or 20 sec has elapsed:
                //std::cout << "TestPower t1 = "<< t1.time << std::endl;
        if( (ret = GetThreshRPM((int)thresh1, &t1, 0, &rpm1)) < 0) return ret;
        t2.time = t1.time;

// Loop until rpm > thresh2 or 2 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh2, &t2, 1, &rpm2)) < 0) return ret;

// Compute acceleration result:
        if( (t2.time -= t1.time) == 0)
                        accel = 0;
        else
                        accel = ((rpm2 - rpm1))/((float)t2.time);

                std::cout << "Finished Power_Test(4)" << std::endl;
        return 0;
    }

    // Run the deceleration portion of the test:
    else if(step == 5)
    {   // 2 sec to get started
                _ftime(&t2);
t2.time += 2;

                // Make sure engine is still running with rpm > thresh2 & get start time &
            // rpm:
        /* for(;;)
                                                        */
while(frame->getRPM()<thresh2)
        {   //Removed for demo
   /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;                   */
   update_clients("RPM Too Low");
                        _ftime(&t1);
   if( t1.time > t2.time) return TIMEOUT;
        }
        rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];
        /* if( (Value[0][0] = rpm1) < thresh2) return INDETERM;
     */
if( rpm1 < thresh2) return INDETERM;

// Loop until rpm < thresh2 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh2, &t1, 2, &rpm1)) < 0) return ret;
        t2.time = t1.time;
        /* rpm1 = Value[0][0];                                                  */

// Loop until rpm < thresh1 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh1, &t2, 2, &rpm2)) < 0) return ret;


// Compute deceleration result:
        if( (t2.time -= t1.time) == 0) return BAD_DATA;
```

```
            /* accel += ((rpm1 - Value[0][0]) * 1000.)/t2;
                  */
            accel += ((rpm1 - rpm2))/((float)t2.time);

      // Finish up:
      /* Next 3 lines modified for demo                                    */
            /* Value[0][0] = accel;                                       */
            /* if(VIDdata[vid].pcf) Value[0][1] = accel/VIDdata[vid].pcf; */
            /* else Value[0][1] = 0;                                      */

      baseline = (accel + baseline)/2;
      pcf = 100*accel/baseline;
      update_clients("",accel,pcf);


                  // Next line removed for demo
                  //LoadDisplayBuff(4, 0, 2);
                  std::cout << "Finished Power_Test(5)" << std::endl;
            }

            std::cout << "Exiting Power_Test" << std::endl;
            return 0;
}/* End of test_power function from class SPowerTest
                  */

/*************************************************************************/
/* Class          SPowerTest
                                                      */
/* Function             set_running
                                                      */
/* Description    Switches vehicle on or off
                  */
/* Date                 8/16/2000
                                                      */
/* Modification
                                                      */
/* References
                                                      */
/*-----------------------------------------------------------------------*/
/* PreCond              none
                                                      */
/*-----------------------------------------------------------------------*/
/* PostCond             frame.StartStop == !frame.StartStop
                              */
/*************************************************************************/
void SPowerTest::set_running(CORBA::Boolean running)
{     // Determine if turning On or Off the vehicle
                  */
      std::cout << "Starting SetRunning..." << std::endl;
      if(!frame->getStartStop())
  {  /* Set the vehicle to running
                  */
            std::cout << "SetRunning:Starting Vehicle" << std::endl;
            frame->setLabelOnOff("ON");
            frame->setRPM(frame->getAccelerator()*65 + 650);
            frame->setjLabel4(frame->getRPM());
```

```
                frame->setStartStop(true);
    }
    else
    {       /* Set the vehicle to off                                          */
                std::cout << "SetRunning:Stopping Vehicle" << std::endl;
                frame->setLabelOnOff("OFF");
      frame->setRPM(0);
      frame->setjLabel4(frame->getRPM());
                frame->setStartStop(false);
    }/* End if                                                                 */

    update_clients();
        std::cout << "Exiting SetRunning..." << std::endl;
}/* End of function set_running from class SPowerTest                         */


/*************************************************************************/
/* Class            SPowerTest                                           */
/* Function                 set_acc                                      */
/* Description      Sets the vehicle accelerator to input value          */
/* Date                     8/16/2000                                    */
/* Modification                                                          */
/* References                                                            */
/*-----------------------------------------------------------------------*/
/* PreCond                  none                                         */
/*-----------------------------------------------------------------------*/
/* PostCond                 Accelerator = acc                            */
/*                  if vehicle is on RPM = Accelerator*65+650 else RPM = 0*/
/*************************************************************************/
void SPowerTest::set_acc(short acc)
{       /* Set the accelerator
                                                */
        std::cout << "Starting SetAcc..." << std::endl;
        frame->setAccelerator(acc);

        /* Check if the vehicle is running or not
                */
        if(frame->getStartStop())
        {       frame->setRPM(frame->getAccelerator()*65+650);
                std::cout << "Setting ACC to " << ((frame->getAccelerator())*65+650) <<
            std::endl;
        }
  else
  {     frame->setRPM(0);
                std::cout << "Setting ACC to 0" << std::endl;
        }

        /* Set the rpm label                                               */
        frame->setjLabel4(frame->getRPM());

        update_clients();
        std::cout << "Exiting SetAcc..." << std::endl;
}/* End of function set_acc from Class SPowerTest                          */

/*************************************************************************/
/* Class            SPowerTest                                           */
```

```
/* Function                set_baseline NOTE: Not Currently Used                    */
/* Description     Sets the vehicle baseline acceleration value                     */
/* Date                    8/16/2000                                                */
/* Modification                                                                     */
/* References                                                                       */
/*--------------------------------------------------------------------------------*/
/* PreCond                 test_power must be run at least once
                           */
/*--------------------------------------------------------------------------------*/
/* PostCond                baseline = acceleration                                  */
/*********************************************************************************/
CORBA::Boolean SPowerTest::set_baseline(float acc)
{  std::cout << "Starting SetBaseline..." << std::endl;
        if(acc>0)
   {  baseline = acc;
      return true;
   }

        std::cout << "Starting SetBaseline..." << std::endl;
        return false;
}/* End of function set_baseline of class SPowerTest                              */

/*********************************************************************************/
/* Class           SPowerTest                                                     */
/* Function                get_index                                              */
/* Description     Returns the next client index (num. of clients +1)             */
/* Date                    8/16/2000                                              */
/* Modification                                                                   */
/* References                                                                     */
/*--------------------------------------------------------------------------------*/
/* PreCond                 none                                                   */
/*--------------------------------------------------------------------------------*/
/* PostCond                none
                                                                                  */
/*********************************************************************************/
short SPowerTest::get_index()
{       std::cout << "Returning index to client " << clnum+1 <<"..."<< std::endl;
        return (short)(clnum+1);
}/* End of function get_index of classs SpowerTest                                */


/*********************************************************************************/
/* Class           SPowerTest
                                                                                  */
/* Function                add_client
                                                                                  */
/* Description     Adds a client to the end of the client list
        */
/* Date                    8/16/2000
                                                                                  */
/* Modification
                                                                                  */
/* References
                                                                                  */
/*--------------------------------------------------------------------------------*/
```

```
/* PreCond                  ior is a valid IOR reference
                                        */
/*-------------------------------------------------------------------------*/
/* PostCond                 client with IOR will be at end of client list
        */
/*************************************************************************/
CORBA::Boolean SPowerTest::add_client(const char * ior,  /* An stringified IOR reference
        */
                                                        const char * name) /* A name for the client
                                        */
{       VehicleModule::vehicleClient_var client;  /* The cleints server                    */
  cnode *temp,  /* Temporarily holds the new client                              */
                *end;  /* Holds the last cnode in client list                    */
        int argv=1;


        std::cout << "Starting AddClient ..."<< std::endl;
  try{
    //Initialize the ORB
                char *jnk[1] = {"SServer"};
                std::cout << "Initializing AddClient ORB..."<< std::endl;
                CORBA::ORB_var orb = CORBA::ORB_init(argv, jnk, "");

    //obtain the root context
                std::cout << "Narrowing AddClient ORB..."<< std::endl;
                CORBA::Object_var obj = orb->string_to_object(ior);
                client = VehicleModule::vehicleClient::_narrow(obj);

        /* If the client list is empty                                                  */
    if(clist == NULL)
    {   /* Set new node to clist                                                        */
                        std::cout << "CList Empty Adding New CLient..."<< std::endl;
                        clist = new cnode;
                        clist->vclient = client;
                        strcpy(clist->name, name);
    }
    else
    {   /* Else put new client at end of list                                           */
                        std::cout << "Clist Not empty Adding New Client..."<< std::endl;
                temp = new cnode;               /* Create a new node                */
                        temp->vclient = client;   /* Add client to new node         */
                        strcpy(temp->name, name);       /* Add client name to new node
                                                        */

                        end = clist;                            /* Set end to first node in list
                                                */
                        while(end->next != NULL)   /* walk through list
                                                        */
                                end = end->next;

      end->next = temp;         /* Add temp to end of list      */
    }/* End if                                                                       */

    // Determine what type of client it is...
          std::cout << "Checking Client Type..."<< std::endl;
    if(name[0] == 'v')
```

```
    {  /* Vehicle Client                                                    */
                    clnum++;
      /* If client is first vehicle to connect return true
        */
                    if(clnum == 1) return true;
    }
    else
    {  /* IETM Client                                                      */
                    ienum++;
                    /* If client is first IETM to connect return true
       */
      if(ienum == 1) return true;
    }/* End if                                                            */
  }
  catch (CORBA::Exception &)
        {       std::cerr << "CORBA exception raised!" << std::endl;
        }

        /* Else client isn't first of its type to conect, return false
        */
        std::cout << "Exiting AddClient..."<< std::endl;
  return true;
}/* End of function add_client of class SPowerTest                       */


/*********************************************************************/
/* Class          SPowerTest                                         */
/* Function               update_clients                             */
/* Description    Performs callback operation on all clients in client list.  */
/* Date                   8/16/2000                                  */
/* Modification                                                      */
/* References                                                        */
/*-------------------------------------------------------------------*/
/* PreCond                none                                       */
/*-------------------------------------------------------------------*/
/* PostCond               Client RPM will be updated                 */
/*********************************************************************/
void SPowerTest::update_clients()
{
  int i = 0;       /* A simple counter                               */
  cnode *tback = NULL; /* A temporary client pointer                 */
  cnode *temp = clist; /* A temporary client pointer                 */

        /* Loop through the client list                               */
        std::cout << "Starting UpdateClients()..."<< std::endl;
  while(temp != NULL)
  { try
            {       /* try to update the client                      */
                    std::cout << "Updating Client "<< i << " with RPM=" << (frame-
                  >getRPM()) << std::endl;
    temp->vclient->set_rpm((short)(frame->getRPM()));
    }
    catch (CORBA::Exception &)
            {       /* If the client didn't respond, remove it from client list
      */
                    if(temp->name[0] == 'v')
```

```
            --clnum;
          else
            --ienum;

          if(tback != NULL)
                              {
            tback->next = temp->next;
            temp = NULL;
            temp = tback;
                              }
                              else
                              {
            clist = temp->next;
            temp = NULL;
            temp = clist;
            continue;
          }/* End if                                                    */
        }/* End try/catch                                              */

        tback = temp;
        temp = temp->next;
        ++i;
    }/* End while                                                      */
    std::cout << "Exiting UpdateClients()..."<< std::endl;
}/* End of function update_clients of class SPOwerTest                */

/*********************************************************************/
/* Class          SPowerTest                                         */
/* Function       update_clients (parameter)                         */
/* Description    Updates the RPM and parameters for all Clients     */
/* Date           8/16/2000                                                 */
/* Modification                                                      */
/* References     update_clients()                                   */
/*-------------------------------------------------------------------*/
/* PreCond        none                                                      */
/*-------------------------------------------------------------------*/
/* PostCond       all clients will have current RPM                  */*/
/*********************************************************************/
void SPowerTest::update_clients(char p[]) /* Paramater               */
{
    int i = 0;      /* A simple counter                              */
    cnode *tback = NULL; /* A temporary client node                 */
    cnode *temp = clist; /* A temporary client node                 */

        std::cout << "Starting UpdateClients(Char)..."<< std::endl;
    while(temp != NULL)
    {  try
                {       /* update the clients                                  */
                        std::cout << "Updating Client..."<< i << std::endl;
       (temp->vclient)->set_rpm((short)frame->getRPM());

                        /* If the cleint is an IETM, update the test parameter */
                        if(temp->name[0] == 'I')
       (temp->vclient)->set_testparm(p);
     }
     catch (CORBA::Exception &)
```

```
                {       /* A client can not be contacted and will be removed
                 */
                if(temp->name[0] == 'v')
                  --clnum;
                else
                  --ienum;

                if(tback != NULL)
                                {
                  tback->next = temp->next;
                  temp = NULL;
                  temp = tback;
                }
                else
                                {
                  clist = temp->next;
                  temp = NULL;
                  temp = clist;
                  continue; /*Continue while                                    */
                }/* End if                                                      */
              }/* End try/catch                                               */

              tback = temp;
              temp = temp->next;
              ++i;
            }/* End while                                                      */
                std::cout << "Exiting UpdateClients(char)..."<< std::endl;
         }/* End of function update_clients of class SPowerTest              */

         /*************************************************************************/
         /* Class          SPowerTest                                          */
         /* Function               update_clients (acceleration, percentage)   */
         /* Description    Updates RPM for all clients and returns acceleration and percentage */
         /*                                      to IETM clients                */
         /* Date                   8/16/2000                                   */
         /* Modification                                                       */
         /* References     update_clients()                                    */
         /*-----------------------------------------------------------------------*/
         /* PreCond                none
                                                                    */
         /*-----------------------------------------------------------------------*/
         /* PostCond               All clients will have an updated RPM
                                   */
         /*                               IETM clients will have an updated acceleration & percentage result
                 */
         /*************************************************************************************/
         void SPowerTest::update_clients(float a,   /* Acceleration from test_power*/
                                         float pcf) /* Percentage from test_power      */
         {
            int i = 0;                          /* A simple counter                  */
            cnode *tback = NULL; /* A temporary client node                       */
            cnode *temp = clist; /* A temporary client node                       */

                std::cout << "Starting UpdateClients(acc, pcf)..."<< std::endl;
            while(temp != NULL)
            {  try
```

```
                {       /* Update the client                                        */
                        std::cout << "Updating Client..."<< i << std::endl;
        (temp->vclient)->set_rpm((short)frame->getRPM());
                        /* If it's an IETM update the results                       */
        if(temp->name[0] == 'I')
                                (temp->vclient)->results(a, pcf);
      }
      catch (CORBA::Exception &)
                { /* The client could not be contacted and will be removed    */
        if(temp->name[0] == 'v')
          --clnum;
        else
          --ienum;

        if(tback != NULL)
                        {
          tback->next = temp->next;
          temp = NULL;
          temp = tback;
                        }
        else
                        {
          clist = temp->next;
          temp = NULL;
          temp = clist;
          continue;
        }/* End if                                                              */
                }/* End try/catch                                               */

      tback = temp;
      temp = temp->next;
      ++i;
    }/* End while                                                               */
        std::cout << "Exiting UpdateClients(acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest                         */

/*************************************************************************/
/* Class        SPowerTest                                                */
/* Function             update_clients (parameter, acceleration, percentage)  */
/* Description  Updates RPM for all clients and returns acceleration and percentage */
/*                                      and parameter to IETM clients        */
/* Date                 8/16/2000                                            */
/* Modification                                                              */
/* References   update_clients()                                             */
/*-------------------------------------------------------------------------*/
/* PreCond              none                                                 */
/*-------------------------------------------------------------------------*/
/* PostCond             All clients will have an updated RPM                 */
/*              IETM clients will have an updated parameter, acceleration &  */
/*                                      percentage result                    */
/*************************************************************************/
void SPowerTest::update_clients(char p[],  /* Parameter from test_power      */
                                float a,   /* Acceleration from test_pwer */
                                float pcf) /* Percentage from test_power     */
{
   int i = 0;       /* A simple counter                                      */
```

```
    cnode *tback = NULL; /* A temporary client node                          */
    cnode *temp = clist; /* A temporary client node                          */

          std::cout << "Starting UpdateClients(Char, acc, pcf)..."<< std::endl;
    while(temp != NULL)
    {  try
                    {       /* Update the client                                    */
                            std::cout << "Updating Client..."<< i << std::endl;
       (temp->vclient)->set_rpm((short)frame->getRPM());
                            /* If it's an IETM client update parameter and results
        */
       if(temp->name[0] == 'I')
       { (temp->vclient)->set_testparm(p);
         (temp->vclient)->results(a,pcf);
                            }/* End if                                           */
      }
      catch (CORBA::Exception &)
                    {       /* The client did not respond so remove it from the list  */
       if(temp->name[0] == 'v')
         --clnum;
       else
         --ienum;

       if(tback != NULL){
         tback->next = temp->next;
         temp = NULL;
         temp = tback;
       }
       else{
         clist = temp->next;
         temp = NULL;
         temp = clist;
         continue;
       }/* End if                                                          */
      }/* End try/catch                                                    */

      tback = temp;
      temp = temp->next;
      ++i;
    }/* End while                                                          */
          std::cout << "Exiting UpdateClients(char, acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest                    */
```

Appendix L:

CORBA Power Test Demonstration Software, C++ Server, for Linux

vehiclemodule.idl:

```
module VehicleModule
{  interface vehicleServer
        {       long test_power(in short step);
                void set_running(in boolean running);
                void set_acc(in short acc);
                boolean set_baseline(in float acc);
                short get_index();
                boolean add_client(in string ior, in string name);
        };

        interface vehicleClient
        {       void set_rpm(in short rpm);
                void set_testparm(in string parm);
                void results(in float accel, in float pcf);
        };
};
```

powertypes.h:

```
/*************************************************************************/
/* File            SPowerTest.cpp                                        */
/* Class  SPowerTest                                                     */
/*************************************************************************/
#include <stdio.h>
#include "powertypes.h"
#include "/root/TAO_ORB/ACE_wrappers/TAO/orbsvcs/orbsvcs/CosNamingC.h"
#include "vehiclemoduleC.h"
#include "vehiclemoduleS.h"
#include <iostream>
#include <time.h>

/*************************************************************************/
/* Class           SPowerTest                                            */
/* Function                                                              */
/* Description                                                           */
/* Date                    8/16/2000                                     */
/* Modification                                                          */
/* References                                                            */
/*-----------------------------------------------------------------------*/
/* PreCond                                                               */
/*-----------------------------------------------------------------------*/
/* PostCond                                                              */
/*************************************************************************/
SPowerTest::SPowerTest()
{  frame = new VehicleFrame;
  clindx = 0;
  clist=NULL;
}/* End of SPowerTest Class Constructor                                  */

/*************************************************************************/
/* Class           SPowerTest                                            */
/* Function                                                              */
```

```
/* Description                                                              */
/* Date                 8/16/2000                                           */
/* Modification                                                             */
/* References                                                               */
/*--------------------------------------------------------------------------*/
/* PreCond                                                                  */
/*--------------------------------------------------------------------------*/
/* PostCond                                                                 */
/****************************************************************************/
int SPowerTest::GetThreshRPM(int thresh, time_t  *timein, int step, float *rpm)
{       time_t currtime,  /* The current system time                        */
            timeout;   /* The time at which a timeout occurs  */

        /* Check to see if in accel or decel portion of test                */
        if(step == 2)
        {       /* In deceleration portion of test timeout = 3 seconds      */
                //std::cout << "ThreshRPM step 2 started..." << std::endl;
                timeout = (*timein) + 3;
                //std::cout << "ThreshRPM step 2 timeout = " << timeout.time << std::endl;
                update_clients("Decelerate");
        }
        else
        {       /* In Acceleration portion of test                          */
                update_clients("Accelerate");
                if(step == 0)
                {       std::cout << "ThreshRPM step 0 started..." << std::endl;
                        /* Start of accel portion of test, timout = 20 seconds  */
                        timeout = (*timein) + 20;
                        //std::cout << "ThreshRPM Timeout+20="<< timeout.time << std::endl;
                        //std::cout << "ThreshRPM time->time = "<< time->time << std::endl;
                        //std::cout << "ThreshRPM time = "<< time << std::endl;
                }
                else
                {       //std::cout << "ThreshRPM step 1 started..." << std::endl;
                        /* Second accel portion of test, timout = 2 seconds  */
                        timeout = (*timein) + 2;
                }
       }/* End if                                                           */


//check for timeout seconds for rpm to exceed thresh
do
{  //std::cout << "ThreshRPM Getting curr time..." << std::endl;
                /* Get the current time and rpm                             */
                currtime = time(NULL);
                //std::cout << "ThreshRPM curr time = "<< currtime.time << std::endl;
                *rpm = frame->getRPM();
                //std::cout << "ThreshRPM RPM = " << frame->getRPM() << std::endl;

                /* Check to see if rpm has met thresholds                   */
                if((step != 2  && *rpm > thresh) || (step == 2 && *rpm < thresh))
   {        //std::cout << "ThreshRPM RPM has met Threshold (" << thresh << //":"<<*rpm <<
            std::endl;
                        /* RPM has met threshold, set time and return ok    */
                //std::cout << "ThreshRPM returning curr time" << std::endl;
                //*timein = currtime;
```

```
                        //std::cout << "ThreshRPM curr time="<<currtime.time << //
        std::endl;
       return 0;
                }
  }while(currtime  < timeout);

       //std::cout << "ThreshRPM Returning Error" << std::endl;
       //std::cout << "ThreshRPM curr time="<<currtime.time<< " timeout="<<timeout.time <<
std::endl;
       /* Timeout has occured so return BAD_DATA error                    */
  return BAD_DATA;
}/* End of function GetThreshRPM of class SPowerTest                              */

/*************************************************************************/
/* Class          SPowerTest                                              */
/* Function               test_power                                      */
/* Description    Measures CI engine power.                               */
/*                                    Test 12 uses the DCA tachometer.    */
/*                                    Test 13 uses the TK tachometer.     */
/* Date                                                                   */
/* Modification   8/16/2000                                               */
/* References     US Army PM-TMDE test_power                              */
/*-----------------------------------------------------------------------*/
/* PreCond    If step = 0, Do nothing                                  */
/*                         1, Set up hardware & parameters for the test.  */
/*                         3, Compute the latest test result while the test is  */
/*                           looping during the idle/governor speed check,  */
/*                         4, Run the acceleration portion of the test.   */
/*                         5, Run the deceleration portion of the test & finish up. */
/*-----------------------------------------------------------------------*/
/* PostCond               Returns:  0 -> test OK                          */
/*                                  -n -> error n                         */
/*************************************************************************/
/*   Notes:                                                               */
/*   Removed lines that called outside functions not available to the demo  */
/*    (i.e. relay setpoints).                                             */
/*   Removed lines that used global variables not pertinent to demo (i.e. DATA)  */
/*   Changed function GetTickCount to system time function                */
/*   moved static variables into class description                        */
/*   changed ovcnt from WORD to int                                       */
/*   Changed Value to Current RPM                                         */
/*         Added rpm2                                                     */
/*************************************************************************/
CORBA::Long SPowerTest::test_power(CORBA::Short step, CORBA::Environment &jj) /* Step that the
test is on              */
{       int ret;  /* The return value if not 0                           */
/*      float *Data[2];                                                   */
// The following added in place of future corba functions
       int tstnum[] = {12,0};
  char Units[4] = {""};
// Use Data to store past results
/*      Data[0] = Value[0];                                               */
/*      Data[1] = Value[1];                                               */
       std::cout << "Test_Power recieved " << step << std::endl;
       // Comm check, no work done:
       if(step == 0) return 0;
```

```
    // Set up for the idle/governor check and/or main test:
    else if(step == 1)
    {       // Demo: Initialize all vars
rpm1 = 0;               // Holds first RPM during test
rpm2 = 0;       // Added to hold second RPM value
pcf = 0;        // Added to hold percentage value
thresh1 = 0;    // Lower RPM Threshold
thresh2 = 0;    // Upper RPM Threshold
accel = 0;      // Acceleration value
ovcnt = 0;              // Counts the number of glitches in Gov

            if(tstnum[0] == 12)
            {       // Next 2 lines removed for demo
                    /* if( (ret = pRelay(c_d, 0, CT1F)) < 0) return ret;       */
                    /* r_p = rev_pulse;                                        */
            }
            else
            {       // Next 2 lines removed for demo
                    /* if( (ret = pRelay(TKrelay[0], 0, CT1F)) < 0) return ret; */
                    /*    r_p = 1.;                                            */
            }

            // Next 2 lines removed for demo
            /* if( (ret = pCounter(0, 1, CG_1M)) < 0) return ret;              */
            /* Mult[0] = r_p * 6.0e4;                                          */

      strcpy(Units,"rpm");
      std::cout << "Finished Power_Test(1)" << std::endl;
      return 0;
    }
    // Compute idle, governor check running test value:
    else if(step == 3)
    {       ret = 0; /* Originally set to InterValue(0);
            */

// Deglitch spikes which get thru tach filter:
      if(ret < 0)
            //Deglitch
  if(++ovcnt < 4) ret = 0;
      else ovcnt = 0;
            std::cout << "Finished Power_Test(3)" << std::endl;
      return ret;
    }

    // Run the acceleration portion of the test:
    else if(step == 4)
    {  // Prepare for Power test:
            if(frame->getvid() == 21 || frame->getvid() == 23)
      {  thresh1 = 2000;
                    thresh2 = 3000;
      }
      else
      {  thresh1 = 1000;
                    thresh2 = 2000;
      }
```

```
            // 20 sec to get started
          t2 = time(NULL);
            //std::cout << "TestPower t2 = "<< t2.time << std::endl;
          t2 += 20;
            //std::cout << "TestPower t2+20 = "<< t2.time << std::endl;
// Make sure engine is still running and get start time & rpm:
            while(frame->getRPM() <= 0)
        {     //Following removed for demo
      /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;                    */
      update_clients("Vehicle Is Off!");
      t1 = time(NULL);
        //std::cout << "TestPower t1 = "<< t1.time << std::endl;
            if( t1 > t2) return TIMEOUT;
        }

       rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];
// Must start with the rpm < thresh1:
       /* if( (Value[0][0] = rpm1) > thresh1) return INDETERM;
     */
if( rpm1 > thresh1) return INDETERM;

            t1 = time(NULL);
// Loop until rpm > thresh1 or 20 sec has elapsed:
            //std::cout << "TestPower t1 = "<< t1.time << std::endl;
       if( (ret = GetThreshRPM((int)thresh1, &t1, 0, &rpm1)) < 0) return ret;
       t2 = t1;
// Loop until rpm > thresh2 or 2 sec has elapsed:
       if( (ret = GetThreshRPM((int)thresh2, &t2, 1, &rpm2)) < 0) return ret;

// Compute acceleration result:
       if( (t2 -= t1) == 0)
                    accel = 0;
       else
                    accel = ((rpm2 - rpm1))/((float)t2);
            std::cout << "Finished Power_Test(4)" << std::endl;
       return 0;
     }

     // Run the deceleration portion of the test:
     else if(step == 5)
     {  // 2 sec to get started
       t2 = time(NULL);//_ftime(&t2);
t2 += 2;

     // Make sure engine is still running with rpm > thresh2 & get start time & rpm:
       /* for(;;)                                                                    */
while(frame->getRPM()<thresh2)
       {  //Removed for demo
   /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;                     */
   update_clients("RPM Too Low");
       t1 = time(NULL);//_ftime(&t1);
   if( t1 > t2) return TIMEOUT;
       }
       rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];
       /* if( (Value[0][0] = rpm1) < thresh2) return INDETERM;
     */
```

```
    if( rpm1 < thresh2) return INDETERM;

    // Loop until rpm < thresh2 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh2, &t1, 2, &rpm1)) < 0) return ret;
        t2 = t1;
        /* rpm1 = Value[0][0];                          */
    // Loop until rpm < thresh1 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh1, &t2, 2, &rpm2)) < 0) return ret;

    // Compute deceleration result:
        if( (t2 -= t1) == 0) return BAD_DATA;
        /* accel += ((rpm1 - Value[0][0]) * 1000.)/t2;
              */
       accel += ((rpm1 - rpm2))/((float)t2);

    // Finish up:
    /* Next 3 lines modified for demo
                        */
        /* Value[0][0] = accel;
                                    */
        /* if(VIDdata[vid].pcf) Value[0][1] = accel/VIDdata[vid].pcf;
      */
        /* else Value[0][1] = 0;
                                  */
    baseline = (accel + baseline)/2;
    pcf = 100*accel/baseline;
    update_clients("",accel,pcf);

            // Next line removed for demo
            //LoadDisplayBuff(4, 0, 2);
            std::cout << "Finished Power_Test(5)" << std::endl;
        }

        std::cout << "Exiting Power_Test" << std::endl;
        return 0;
}/* End of test_power function from class SPowerTest
            */


/*************************************************************************/
/* Class          SPowerTest
                                                    */
/* Function             set_running
                                                    */
/* Description    Switches vehicle on or off
                */
/* Date                 8/16/2000
                                                    */
/* Modification
                                                    */
/* References
                                                    */
/*-----------------------------------------------------------------------*/
/* PreCond              none
                                                    */
/*-----------------------------------------------------------------------*/
```

```
/* PostCond                frame.StartStop == !frame.StartStop
                                                        */
/*************************************************************************/
void SPowerTest::set_running(CORBA::Boolean running, CORBA::Environment &jj)
{       // Determine if turning On or Off the vehicle
                        */
        std::cout << "Starting SetRunning..." << std::endl;
        if(!frame->getStartStop())
  {  /* Set the vehicle to running
                        */
                std::cout << "SetRunning:Starting Vehicle" << std::endl;
                frame->setLabelOnOff("ON");
                frame->setRPM(frame->getAccelerator()*65 + 650);
                frame->setjLabel4(frame->getRPM());
                frame->setStartStop(true);
  }
  else
  {     /* Set the vehicle to off
                                */
                std::cout << "SetRunning:Stopping Vehicle" << std::endl;
                frame->setLabelOnOff("OFF");
    frame->setRPM(0);
    frame->setjLabel4(frame->getRPM());
                frame->setStartStop(false);
  }/* End if                                                               */

  update_clients();
        std::cout << "Exiting SetRunning..." << std::endl;
}/* End of function set_running from class SPowerTest
                        */
/*************************************************************************/
/* Class           SPowerTest
                                                        */
/* Function                set_acc
                                                        */
/* Description     Sets the vehicle accelerator to input value
        */
/* Date                    8/16/2000
                                                        */
/* Modification
                                                        */
/* References
                                                        */
/*-----------------------------------------------------------------------*/
/* PreCond                 none
                                                        */
/*-----------------------------------------------------------------------*/
/* PostCond                Accelerator = acc
                                        */
/*                                 if vehicle is on RPM = Accelerator*65+650 else RPM = 0
                        */
/*************************************************************************/
void SPowerTest::set_acc(CORBA::Short acc, CORBA::Environment &jj)
{       /* Set the accelerator
                                                        */
        std::cout << "Starting SetAcc..." << std::endl;
```

```
        frame->setAccelerator(acc);

        /* Check if the vehicle is running or not
                        */
        if(frame->getStartStop())
        {       frame->setRPM(frame->getAccelerator()*65+650);
                std::cout << "Setting ACC to " << ((frame->getAccelerator())*65+650) << std::endl;
        }
   else
   {    frame->setRPM(0);
                std::cout << "Setting ACC to 0" << std::endl;
        }

        /* Set the rpm label
                                                */
        frame->setjLabel4(frame->getRPM());

        update_clients();
        std::cout << "Exiting SetAcc..." << std::endl;
}/* End of function set_acc from Class SPowerTest
                */
/*************************************************************************/
/* Class            SPowerTest
                                                */
/* Function                 set_baseline NOTE: Not Currently Used
                            */
/* Description      Sets the vehicle baseline acceleration value
        */
/* Date                     8/16/2000
                                                */
/* Modification
                                                */
/* References
                                                */
/*-----------------------------------------------------------------------*/
/* PreCond                  test_power must be run at least once
                            */
/*-----------------------------------------------------------------------*/
/* PostCond                 baseline = acceleration
                                        */
/*************************************************************************/
CORBA::Boolean SPowerTest::set_baseline(CORBA::Float acc, CORBA::Environment &jj)
{  std::cout << "Starting SetBaseline..." << std::endl;
        if(acc>0)
  {  baseline = acc;
     return true;
  }

        std::cout << "Starting SetBaseline..." << std::endl;
        return false;
}/* End of function set_baseline of class SPowerTest
        */
/*************************************************************************/
/* Class            SPowerTest
                                                */
```

```
/* Function                get_index
                                                                */
/* Description     Returns the next client index (num. of clients +1)
        */
/* Date                    8/16/2000
                                                                */
/* Modification
                                                                */
/* References
                                                                */
/*-------------------------------------------------------------------------*/
/* PreCond                 none
                                                                */
/*-------------------------------------------------------------------------*/
/* PostCond                none
                                                                */
/*************************************************************************/
CORBA::Short SPowerTest::get_index(CORBA::Environment &jj)
{       std::cout << "Returning index to client " << clindx+1 <<"..."<< std::endl;
        return (CORBA::Short)(++clindx);
}/* End of function get_index of classs SpowerTest
                */
/*************************************************************************/
/* Class           SPowerTest
                                                        */
/* Function                add_client                      ~~
                                                                */
/* Description     Adds a client to the end of the client list
        */
/* Date                    8/16/2000
                                                                */
/* Modification
                                                                */
/* References
                                                                */
/*-------------------------------------------------------------------------*/
/* PreCond                 ior is a valid IOR reference
                                */
/*-------------------------------------------------------------------------*/
/* PostCond                client with IOR will be at end of client list
        */
/*************************************************************************/
CORBA::Boolean SPowerTest::add_client(const char * ior,  /* An stringified IOR reference
        */
                                                        const char * name,
                                        CORBA::Environment &jj) /* A name for the client
                        */
{       VehicleModule::vehicleClient_var client;  /* The cleints server
                */
  cnode *temp,  /* Temporarily holds the new client
        */
                 *end;   /* Holds the last cnode in client list
                */
        int argv=1;

        std::cout << "Starting AddClient ..."<< std::endl;
```

```
        //   try{
    //Initialize the ORB
                    char *jnk[1] = {"SServer"};
                    std::cout << "Initializing AddClient ORB..."<< std::endl;
                    CORBA::ORB_var orb = CORBA::ORB_init(argv, jnk, "");

    //obtain the root context
                    std::cout << "Narrowing AddClient ORB..."<< std::endl;
                    CORBA::Object_var obj = orb->string_to_object(ior);
                    std::cout << "IOR Converted..." << std::endl;
                    std::cout << "IOR is: " << ior << std::endl;
                    client = VehicleModule::vehicleClient::_narrow(obj);
                    std::cout << "Client Gotten..." << std::endl;

          /* If the client list is empty
                                        */
    if(clist == NULL)
    {    /* Set new node to clist
                                */
                                std::cout << "CList Empty Adding New CLient..."<< std::endl;
                                clist = new cnode();
std::cout << "Got here 0\n";
//                              clist->next = NULL;
std::cout << "Got here 0.5\n";
                                clist->vclient = client;
                                strcpy(clist->name, name);
    }
    else
    {    /* Else put new client at end of list
         */
                                std::cout << "Clist Not empty Adding New Client..."<< std::endl;
                        temp = new cnode();                     /* Create a new node
                                */

                                temp->vclient = client;    /* Add client to new node
        */

                                strcpy(temp->name, name);         /* Add client name to new node
                    */


                                end = clist;                                /* Set end to first node in list
                    */

                                while(end->next != NULL) {    /* walk through list
                                */
                                        end = end->next;
                                }
      end->next = temp;             /* Add temp to end of list                                        */
    }/* End if     */
    // Determine what type of client it is...
          std::cout << "Checking Client Type..."<< std::endl;
    if(name[0] == 'v')
    {   /* Vehicle Client
                                */
                    clnum++;
```

```
        /* If client is first vehicle to connect return true
           */
                        if(clnum == 1) return true;
      }
      else
      {  /* IETM Client
                                                */
                        ienum++;
                        /* If client is first IETM to connect return true
           */
        if(ienum == 1) return true;
      }/* End if
                                                */
      // }
      // catch (CORBA::Exception &)
      //    {        std::cerr << "CORBA exception raised!" << std::endl;
      //    }

          /* Else client isn't first of its type to conect, return false
          */
          std::cout << "Exiting AddClient..."<< std::endl;
    return true;
}/* End of function add_client of class SPowerTest
                */
/*********************************************************************/
/* Class            SPowerTest
                                                */
/* Function                 update_clients
                                                */
/* Description      Performs callback operation on all clients in client list.          */
/* Date                     8/16/2000
                                                */
/* Modification
                                                */
/* References
                                                */
/*-------------------------------------------------------------------------*/
/* PreCond                  none
                                                */
/*-------------------------------------------------------------------------*/
/* PostCond                 Client RPM will be updated
                                                */
/*********************************************************************/
void SPowerTest::update_clients()
{
    int i = 0;          /* A simple counter
                        */
    cnode *tback = NULL; /* A temporary client pointer
                        */
    cnode *temp = clist; /* A temporary client pointer
          */

          /* Loop through the client list
                                        */
          std::cout << "Starting UpdateClients()..."<< std::endl;
    while(temp != NULL)
```

```
    { //try{
                        /* try to update the client
                                        */
                        std::cout << "Updating Client "<< i << " with RPM=" << (frame->getRPM())
<< std::endl;
        temp->vclient->set_rpm((short)(frame->getRPM()));
            //      }
            //catch (CORBA::Exception &)
            //              {       /* If the client didn't respond, remove it from client list
                    */
            //              if(temp->name[0] == 'v')
        //  --clnum;
            // else
        //  --ienum;
            //
        //if(tback != NULL)
            //                          {
            // tback->next = temp->next;
        //  temp = NULL;
        //  temp = tback;
            //              }
            //      else
            //      {
            // clist = temp->next;
            // temp = NULL;
            // temp = clist;
            // continue;
      // }/* End if
                                                    */
  // }/* End try/catch
                                                */

    tback = temp;
    temp = temp->next;
    ++i;
 }/* End while
                                                        */
 std::cout << "Exiting UpdateClients()..."<< std::endl;
}/* End of function update_clients of class SPOwerTest
                */
/*************************************************************************/
/* Class            SPowerTest
                                                    */
/* Function                 update_clients (parameter)
                                */
/* Description      Updates the RPM and parameters for all Clients
        */
/* Date                     8/16/2000
                                                        */
/* Modification
                                                        */
/* References       update_clients()
                                */
/*-------------------------------------------------------------------------*/
/* PreCond                  none
                                                        */
```

```
/*-------------------------------------------------------------------------*/
/* PostCond                    all clients will have current RPM
                               */
/*                                      IETM clients will have updated parameter
                               */
/*************************************************************************/
void SPowerTest::update_clients(char p[]) /* Paramater
                    */
{
   int i = 0;          /* A simple counter
                       */
   cnode *tback = NULL; /* A temporary client node
                       */
   cnode *temp = clist; /* A temporary client node
                       */

         std::cout << "Starting UpdateClients(Char)..."<< std::endl;
   while(temp != NULL)
{  //try{
                               /* update the clients
                                                       */
                               std::cout << "Updating Client..."<< i << std::endl;
       (temp->vclient)->set_rpm((short)frame->getRPM());

                               /* If the cleint is an IETM, update the test parameter
        */
                               if(temp->name[0] == 'I')
        (temp->vclient)->set_testparm(p);
                               // }
                               // catch (CORBA::Exception &)
                               //                      {          /* A client can not be contacted and will be
                                                      removed */
                               // if(temp->name[0] == 'v')
                               // --clnum;
                               // else
                               //         --ienum;

                               //          if(tback != NULL)
                               //          {
                               //        tback->next = temp->next;
                               //  temp = NULL;
                               //   temp = tback;
                               // }
                               // else
                               //          {
                               //  clist = temp->next;
                               //  temp = NULL;
                               //  temp = clist;
                               //   continue; /*Continue while
                                                       */
                               //    }/* End if
                                                                       */

                               //  }/* End try/catch
                                                                       */

     tback = temp;
```

```
    temp = temp->next;
    ++i;
   }/* End while
                                                    */
        std::cout << "Exiting UpdateClients(char)..."<< std::endl;
}/* End of function update_clients of class SPowerTest
                */
/*************************************************************************/
/* Class          SPowerTest
                                                    */
/* Function              update_clients (acceleration, percentage)
                */
/* Description    Updates RPM for all clients and returns acceleration and percentage    */
/*                                        to IETM clients
                                                    */
/* Date                  8/16/2000
                                                    */
/* Modification
                                                    */
/* References     update_clients()
                                */
/*-----------------------------------------------------------------------*/
/* PreCond               none
                                                    */
/*-----------------------------------------------------------------------*/
/* PostCond              All clients will have an updated RPM
                         */
/*                              IETM clients will have an updated acceleration & percentage result
        */
/*************************************************************************/
void SPowerTest::update_clients(float a,  /* Acceleration from test_power                    */
                                                                    float pcf) /* Percentage from
test_power                      */
{
   int i = 0;                    /* A simple counter
                                 */
   cnode *tback = NULL; /* A temporary client node
                */
   cnode *temp = clist; /* A temporary client node
                */

        std::cout << "Starting UpdateClients(acc, pcf)..."<< std::endl;
   while(temp != NULL)
    { // try{
                        /* Update the client
                                                    */
                        std::cout << "Updating Client..."<< i << std::endl;
          (temp->vclient)->set_rpm((short)frame->getRPM());
                /* If it's an IETM update the results
                */
         if(temp->name[0] == 'I')
                        (temp->vclient)->results(a, pcf);
        // }
        // catch (CORBA::Exception &)
        //      {  /* The client could not be contacted and will be removed
        */
```

```
        // if(temp->name[0] == 'v')
        //     --clnum;
        // else
        //     --ienum;
        //
        // if(tback != NULL)
        //                 {
    //     tback->next = temp->next;
    //     temp = NULL;
    //     temp = tback;
        //                 }
    //  else
        //                 {
    //     clist = temp->next;
        //     temp = NULL;
        //     temp = clist;
        //     continue;
    //  }/* End if
                                                    */
        //          }/* End try/catch
                                                    */
        //
    tback = temp;
    temp = temp->next;
    ++i;
   }/* End while
                                                    */
        std::cout << "Exiting UpdateClients(acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest
                */


/*********************************************************************/
/* Class            SPowerTest
                                                    */
/* Function                 update_clients (parameter, acceleration, percentage)
        */
/* Description      Updates RPM for all clients and returns acceleration and percentage     */
/*                                  and parameter to IETM clients
                                    */
/* Date                     8/16/2000
                                                    */
/* Modification
                                                    */
/* References       update_clients()
                                */
/*-------------------------------------------------------------------------*/
/* PreCond                  none
                                                    */
/*-------------------------------------------------------------------------*/
/* PostCond                 All clients will have an updated RPM
                            */
/*                                  IETM clients will have an updated parameter, acceleration &
        */
/*                                      percentage result
                                        */
```

```
/*************************************************************************/
void SPowerTest::update_clients(char p[],  /* Parameter from test_power
        */
                                                                float a,   /* Acceleration from
test_pwer                       */
                                                                float pcf) /* Percentage from
test_power                      */
{
   int i = 0;       /* A simple counter
                    */
   cnode *tback = NULL; /* A temporary client node
                    */
   cnode *temp = clist; /* A temporary client node
                    */

        std::cout << "Starting UpdateClients(Char, acc, pcf)..."<< std::endl;
   while(temp != NULL)
    {  //try{
                        /* Update the client
                                                        */
                std::cout << "Updating Client..."<< i << std::endl;
         (temp->vclient)->set_rpm((short)frame->getRPM());
                /* If it's an IETM client update parameter and results
        */
         if(temp->name[0] == 'T')
         {      (temp->vclient)->set_testparm(p);
            (temp->vclient)->results(a,pcf);
                }/* End if
                                                                */
         //  }
         // catch (CORBA::Exception &)
         // {      /* The client did not respond so remove it from the list
        */
         // if(temp->name[0] == 'v')
         //    --clnum;
         // else
         //    --ienum;
         //
         // if(tback != NULL){
         //    tback->next = temp->next;
         //    temp = NULL;
         //    temp = tback;
         // }
         // else{
         //    clist = temp->next;
         //    temp = NULL;
         //    temp = clist;
         //    continue;
         // }/* End if
                                                        */
         // }/* End try/catch
                                                        */

    tback = temp;
    temp = temp->next;
    ++i;
```

```
    }/* End while
                                                                */
        std::cout << "Exiting UpdateClients(char, acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest
                */
```

sserver.cpp:

```
/*************************************************************************/
/* File             SServer.cpp
                                                                */
/* Class  SServer                                                          */
/*************************************************************************/
#include <stdio.h>
#include "powertypes.h"
#include "/root/TAO_ORB/ACE_wrappers/TAO/orbsvcs/orbsvcs/CosNamingC.h"
#include <iostream>
#include <sys/types.h>
#include <sys/timeb.h>

/*************************************************************************/
/* Function                 mian                                                   */
/* Description      Starts the server                                      */
/* Date                     8/16/2000                                              */
/* Modification
                                                                */
/* References
                                                                */
/*-----------------------------------------------------------------------*/
/* PreCond                  none
                                                                */
/*-----------------------------------------------------------------------*/
/* PostCond                 SServer will be started
                                                        */
/*************************************************************************/
int main(int argc, char *argv[])
{       std::cout << "argc = " << argc << " argv[0] = " << argv[0] << std::endl;
        SServer *s = new SServer(argc, argv);
        return 0;
}/* End of function main
                                                        */

/*************************************************************************/
/* Class            SServer
                                                        */
/* Function                 SServer
                                                                */
/* Description      Constructor creates server and initializes value
        */
/* Date                     8/16/2000
                                                                */
/* Modification
                                                                */
/* References
                                                                */
/*-----------------------------------------------------------------------*/
```

```
/* PreCond                  none
                                                                    */
/*-------------------------------------------------------------------*/
/* PostCond                 SServer will be started and initialized                    */
/*********************************************************************/
SServer::SServer(int argc, char *argv[])
{       //Start up CORBA ORB
 //     try{
        std::cout << "Initializing ORB...\n" << std::endl;
              CORBA::ORB_var orb = CORBA::ORB_init (argc, argv,"");
          CORBA::Object_var poa_object = orb->resolve_initial_references ("RootPOA");

              std::cout << "Initializing POA...\n" << std::endl;
          PortableServer::POA_var poa = PortableServer::POA::_narrow (poa_object.in ());
              PortableServer::POAManager_var poa_manager = poa->the_POAManager ();
              poa_manager->activate();

              // Create the servant
              std::cout << "Creating Servant...\n" << std::endl;
              SPowerTest * servant = new SPowerTest();

              // Activate it to obtain the object reference
              std::cout << "Activating Servant...\n" << std::endl;
              VehicleModule::vehicleServer_var myobject=(*servant)._this();

              // Get the Naming Context reference
              std::cout << "Getting Naming Service...\n" << std::endl;
              CORBA::Object_var naming_context_object =
              orb->resolve_initial_references ("NameService");
              CosNaming::NamingContext_var naming_context =
              CosNaming::NamingContext::_narrow (naming_context_object.in ());

              // Create and initialize the name.
              CosNaming::Name name (1);
              name.length (1);
              name[0].id = CORBA::string_dup ("Vehicle");

              // Bind the object
              std::cout << "Naming Context Bound...\n" << std::endl;
              naming_context->bind (name, myobject.in ());

              std::cout << "Server is running... \nStart clients now..." << std::endl;
              orb->run ();
              std::cout << "Orb just ran" << std::endl;
              // Destroy the POA, waiting until the destruction terminates
//            poa->destroy (1, 1);
//            orb->destroy ();
              //        }
        //    /*        catch (CORBA::Exception &)
        //{     std::cerr << "CORBA exception raised!" << std::endl;
        //}*/
}/* End of constructor SServer::SServer
                                */
```

spowertest.cpp:

```
/*********************************************************************/
/* File             SPowerTest.cpp                                   */
                                                                    */
```

```
/* Class SPowerTest                                                    */
/*********************************************************************/
#include <stdio.h>
#include "powertypes.h"
#include "/root/TAO_ORB/ACE_wrappers/TAO/orbsvcs/orbsvcs/CosNamingC.h"
#include "vehiclemoduleC.h"
#include "vehiclemoduleS.h"
#include <iostream>
#include <time.h>

/*********************************************************************/
/* Class          SPowerTest                                         */
/* Function                                                          */
/* Description                                                       */
/* Date                   8/16/2000                                  */
/* Modification                                                      */
/* References                                                        */
/*-------------------------------------------------------------------*/
/* PreCond                                                           */
/*-------------------------------------------------------------------*/
/* PostCond                                                          */
/*********************************************************************/
SPowerTest::SPowerTest()
{  frame = new VehicleFrame;
  clindx = 0;
  clist=NULL;
}/* End of SPowerTest Class Constructor                              */

/*********************************************************************/
/* Class          SPowerTest                                         */
/* Function                                                          */
/* Description                                                       */
/* Date                   8/16/2000                                  */
/* Modification                                                      */
/* References                                                        */
/*-------------------------------------------------------------------*/
/* PreCond                                                           */
/*-------------------------------------------------------------------*/
/* PostCond                                                          */
/*********************************************************************/
int SPowerTest::GetThreshRPM(int thresh, time_t  *timein, int step, float *rpm)
{       time_t currtime,  /* The current system time                 */
            timeout;   /* The time at which a timeout occurs  */

        /* Check to see if in accel or decel portion of test         */
        if(step == 2)
        {       /* In deceleration portion of test timeout = 3 seconds */
                //std::cout << "ThreshRPM step 2 started..." << std::endl;
                timeout = (*timein) + 3;
//std::cout << "ThreshRPM step 2 timeout = " << timeout.time << std::endl;
                update_clients("Decelerate");
        }
        else
        {       /* In Acceleration portion of test                   */
                update_clients("Accelerate");
                if(step == 0)
```

```
                {       std::cout << "ThreshRPM step 0 started..." << std::endl;
                        /* Start of accel portion of test, timout = 20 seconds      */
                        timeout = (*timein) + 20;
        //std::cout << "ThreshRPM Timeout+20="<< timeout.time << std::endl;
        //std::cout << "ThreshRPM time->time = "<< time->time << std::endl;
        //std::cout << "ThreshRPM time = "<< time << std::endl;
                }
                else
                {       //std::cout << "ThreshRPM step 1 started..." << std::endl;
                        /* Second accel portion of test, timout = 2 seconds       */
                        timeout = (*timein) + 2;
                }
        }/* End if                                                                */


    //check for timeout seconds for rpm to exceed thresh
    do
    {   //std::cout << "ThreshRPM Getting curr time..." << std::endl;
                /* Get the current time and rpm                                   */
                currtime = time(NULL);
                //std::cout << "ThreshRPM curr time = "<< currtime.time << std::endl;
                *rpm = frame->getRPM();
                //std::cout << "ThreshRPM RPM = " << frame->getRPM() << std::endl;

                /* Check to see if rpm has met thresholds                         */
                if((step != 2  && *rpm > thresh) || (step == 2 && *rpm < thresh))
                {       //std::cout << "ThreshRPM RPM has met Threshold (" << thresh << ":"<<*rpm
                        << std::endl;
                        /* RPM has met threshold, set time and return ok          */
                        //std::cout << "ThreshRPM returning curr time" << std::endl;
                        *timein = currtime;
//std::cout << "ThreshRPM curr time="<<currtime.time << std::endl;
        return 0;
                }
    }while(currtime  < timeout);

        //std::cout << "ThreshRPM Returning Error" << std::endl;
        //std::cout << "ThreshRPM curr time="<<currtime.time<< " timeout="<<timeout.time <<
std::endl;
        /* Timeout has occured so return BAD_DATA error                           */
    return BAD_DATA;
}/* End of function GetThreshRPM of class SPowerTest                           */

/*************************************************************************/
/* Class            SPowerTest                                           */
/* Function                 test_power                                    */
/* Description      Measures CI engine power.                            */
/*                                      Test 12 uses the DCA tachometer.  */
/*                                      Test 13 uses the TK tachometer.   */
/* Date                                                                  */
/* Modification     8/16/2000                                            */
/* References       US Army PM-TMDE test_power                           */
/*-----------------------------------------------------------------------*/
/* PreCond      If step = 0, Do nothing                                  */
/*                      1, Set up hardware & parameters for the test.    */
/*                      3, Compute the latest test result while the test is */
```

```
/*                              looping during the idle/governor speed check,      */
/*                            4, Run the acceleration portion of the test.         */
/*                            5, Run the deceleration portion of the test & finish up. */
/*-------------------------------------------------------------------------*/
/* PostCond                Returns:  0 -> test OK                                  */
/*                                   -n -> error n                                 */
/*************************************************************************/
/*   Notes:                                                                        */
/*   Removed lines that called outside functions not available to the demo         */
/*    (i.e. relay setpoints).                                                      */
/*   Removed lines that used global variables not pertinent to demo (i.e. DATA)    */
/*   Changed function GetTickCount to system time function                         */
/*   moved static variables into class description                                */
/*   changed ovcnt from WORD to int                                                */
/*   Changed Value to Current RPM                                                  */
/*         Added rpm2                                                              */
/*************************************************************************/
CORBA::Long SPowerTest::test_power(CORBA::Short step, CORBA::Environment &jj) /* Step that the
test is on          */
{       int ret;  /* The return value if not 0                                   */
/*      float *Data[2];                                                           */
// The following added in place of future corba functions
        int tstnum[] = {12,0};
  char Units[4] = {""};
// Use Data to store past results
/*      Data[0] = Value[0];                                                       */
/*      Data[1] = Value[1];                                                       */

        std::cout << "Test_Power recieved " << step << std::endl;
        // Comm check, no work done:
        if(step == 0) return 0;

        // Set up for the idle/governor check and/or main test:
        else if(step == 1)
        {       // Demo: Initialize all vars
    rpm1 = 0;               // Holds first RPM during test
    rpm2 = 0;     // Added to hold second RPM value
    pcf = 0;      // Added to hold percentage value
    thresh1 = 0;   // Lower RPM Threshold
    thresh2 = 0;   // Upper RPM Threshold
    accel = 0;    // Acceleration value
    ovcnt = 0;              // Counts the number of glitches in Gov

                if(tstnum[0] == 12)
                {       // Next 2 lines removed for demo
                        /* if( (ret = pRelay(c_d, 0, CT1F)) < 0) return ret;     */
          /* r_p = rev_pulse;                                                    */
                }
                else
                {       // Next 2 lines removed for demo
                        /* if( (ret = pRelay(TKrelay[0], 0, CT1F)) < 0) return ret; */
                        /*    r_p = 1.;                                        */
                }

                // Next 2 lines removed for demo
                /* if( (ret = pCounter(0, 1, CG_1M)) < 0) return ret;           */
```

```
            /* Mult[0] = r_p * 6.0e4;                                    */

        strcpy(Units,"rpm");
        std::cout << "Finished Power_Test(1)" << std::endl;
        return 0;
      }

      // Compute idle, governor check running test value:
      else if(step == 3)
      {       ret = 0; /* Originally set to InterValue(0);
              */

 // Deglitch spikes which get thru tach filter:
        if(ret < 0)
              //Deglitch
    if(++ovcnt < 4) ret = 0;
        else ovcnt = 0;

              std::cout << "Finished Power_Test(3)" << std::endl;
        return ret;
      }

      // Run the acceleration portion of the test:
      else if(step == 4)
      {  // Prepare for Power test:
              if(frame->getvid() == 21 || frame->getvid() == 23)
        {  thresh1 = 2000;
                    thresh2 = 3000;
        }
        else
        {  thresh1 = 1000;
                    thresh2 = 2000;
        }

            // 20 sec to get started
          t2 = time(NULL);
            //std::cout << "TestPower t2 = "<< t2.time << std::endl;
          t2 += 20;
            //std::cout << "TestPower t2+20 = "<< t2.time << std::endl;

// Make sure engine is still running and get start time & rpm:
              while(frame->getRPM() <= 0)
        {     //Following removed for demo
      /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;      */
      update_clients("Vehicle Is Off!");
      t1 = time(NULL);
          //std::cout << "TestPower t1 = "<< t1.time << std::endl;
            if( t1 > t2) return TIMEOUT;
        }

        rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];

// Must start with the rpm < thresh1:
        /* if( (Value[0][0] = rpm1) > thresh1) return INDETERM;
     */
if( rpm1 > thresh1) return INDETERM;
```

```
            t1 = time(NULL);
// Loop until rpm > thresh1 or 20 sec has elapsed:
            //std::cout << "TestPower t1 = "<< t1.time << std::endl;
        if( (ret = GetThreshRPM((int)thresh1, &t1, 0, &rpm1)) < 0) return ret;
        t2 = t1;

// Loop until rpm > thresh2 or 2 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh2, &t2, 1, &rpm2)) < 0) return ret;

// Compute acceleration result:
        if( (t2 -= t1) == 0)
                        accel = 0;
        else
                        accel = ((rpm2 - rpm1))/((float)t2);

            std::cout << "Finished Power_Test(4)" << std::endl;
        return 0;
    }

    // Run the deceleration portion of the test:
    else if(step == 5)
    {  // 2 sec to get started
      t2 = time(NULL);//_ftime(&t2);
t2 += 2;

            // Make sure engine is still running with rpm > thresh2 & get start time & rpm:
        /* for(;;)
                                                    */
while(frame->getRPM()<thresh2)
        {  //Removed for demo
   /* if( (ret = pExecuteTests(0, Nsamp, Data)) < 0) return ret;                    */
   update_clients("RPM Too Low");
        t1 = time(NULL);//_ftime(&t1);
   if( t1 > t2) return TIMEOUT;
        }
        rpm1 = frame->getRPM();//(rev_pulse * 6.0e4)/Value[0][0];
        /* if( (Value[0][0] = rpm1) < thresh2) return INDETERM;
     */
if( rpm1 < thresh2) return INDETERM;

// Loop until rpm < thresh2 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh2, &t1, 2, &rpm1)) < 0) return ret;
        t2 = t1;
        /* rpm1 = Value[0][0];                                                    */
// Loop until rpm < thresh1 or 3 sec has elapsed:
        if( (ret = GetThreshRPM((int)thresh1, &t2, 2, &rpm2)) < 0) return ret;

// Compute deceleration result:
        if( (t2 -= t1) == 0) return BAD_DATA;
        /* accel += ((rpm1 - Value[0][0]) * 1000.)/t2;
                */
     accel += ((rpm1 - rpm2))/((float)t2);

// Finish up:
/* Next 3 lines modified for demo                                               */
```

```
            /* Value[0][0] = accel;                                          */
            /* if(VIDdata[vid].pcf) Value[0][1] = accel/VIDdata[vid].pcf;    */
            /* else Value[0][1] = 0;                                         */
        baseline = (accel + baseline)/2;
        pcf = 100*accel/baseline;
        update_clients("",accel,pcf);
                    // Next line removed for demo
                    //LoadDisplayBuff(4, 0, 2);
                    std::cout << "Finished Power_Test(5)" << std::endl;
            }

            std::cout << "Exiting Power_Test" << std::endl;
            return 0;
    }/* End of test_power function from class SPowerTest                    */


    /*************************************************************************/
    /* Class            SPowerTest
                                                                    */
    /* Function                 set_running
                                                                    */
    /* Description      Switches vehicle on or off
                                                                    */
    /* Date                     8/16/2000
                                                                    */
    /* Modification
                                                                    */
    /* References
                                                                    */
    /*-----------------------------------------------------------------------*/
    /* PreCond                  none
                                                                    */
    /*-----------------------------------------------------------------------*/
    /* PostCond                 frame.StartStop == !frame.StartStop
                                                                    */
    /*************************************************************************/
    void SPowerTest::set_running(CORBA::Boolean running, CORBA::Environment &jj)
    {       // Determine if turning On or Off the vehicle
                    */
            std::cout << "Starting SetRunning..." << std::endl;
            if(!frame->getStartStop())
      {  /* Set the vehicle to running
                                    */
                    std::cout << "SetRunning:Starting Vehicle" << std::endl;
                    frame->setLabelOnOff("ON");
                    frame->setRPM(frame->getAccelerator()*65 + 650);
                    frame->setjLabel4(frame->getRPM());
                    frame->setStartStop(true);
      }
      else
      {     /* Set the vehicle to off
                                        */
                    std::cout << "SetRunning:Stopping Vehicle" << std::endl;
                    frame->setLabelOnOff("OFF");
        frame->setRPM(0);
        frame->setjLabel4(frame->getRPM());
```

```
                frame->setStartStop(false);
    }/* End if
                                                                */

    update_clients();
        std::cout << "Exiting SetRunning..." << std::endl;
}/* End of function set_running from class SPowerTest                    */

/*************************************************************************/
/* Class            SPowerTest
                                                                */
/* Function                 set_acc
                                                                */
/* Description      Sets the vehicle accelerator to input value                 */
/* Date                     8/16/2000                                          */
/* Modification                                                                */
/* References                                                                  */
/*-------------------------------------------------------------------------*/
/* PreCond                  none
                                                                */
/*-------------------------------------------------------------------------*/
/* PostCond                 Accelerator = acc                                  */
/*                  if vehicle is on RPM = Accelerator*65+650 else RPM = 0      */
/*************************************************************************/
void SPowerTest::set_acc(CORBA::Short acc, CORBA::Environment &jj)
{       /* Set the accelerator
                                                                */
        std::cout << "Starting SetAcc..." << std::endl;
        frame->setAccelerator(acc);

        /* Check if the vehicle is running or not                             */
        if(frame->getStartStop())
        {       frame->setRPM(frame->getAccelerator()*65+650);
                std::cout << "Setting ACC to " << ((frame->getAccelerator())*65+650) << std::endl;
        }
  else
  {     frame->setRPM(0);
                std::cout << "Setting ACC to 0" << std::endl;
        }

        /* Set the rpm label        */
        frame->setjLabel4(frame->getRPM());

        update_clients();
        std::cout << "Exiting SetAcc..." << std::endl;
}/* End of function set_acc from Class SPowerTest   */

/*************************************************************************/
/* Class            SPowerTest
                                                                */
/* Function                 set_baseline NOTE: Not Currently Used
                            */
/* Description      Sets the vehicle baseline acceleration value
        */
/* Date                     8/16/2000
                                                                */
```

```
/* Modification                                                          */
/* References                                                            */
/*-----------------------------------------------------------------------*/
/* PreCond          test_power must be run at least once
                    */
/*-----------------------------------------------------------------------*/
/* PostCond         baseline = acceleration
                                                  */
/************************************************************************/
CORBA::Boolean SPowerTest::set_baseline(CORBA::Float acc, CORBA::Environment &jj)
{  std::cout << "Starting SetBaseline..." << std::endl;
        if(acc>0)
   {  baseline = acc;
      return true;
   }

        std::cout << "Starting SetBaseline..." << std::endl;
        return false;
}/* End of function set_baseline of class SPowerTest */

/************************************************************************/
/* Class        SPowerTest
                                                  */
/* Function         get_index
                                                  */
/* Description  Returns the next client index (num. of clients +1)
        */
/* Date             8/16/2000
                                                  */
/* Modification                                                          */
                                                  */
/* References                                                            */
                                                  */
/*-----------------------------------------------------------------------*/
/* PreCond          none
                                                  */
/*-----------------------------------------------------------------------*/
/* PostCond         none
                                                  */
/************************************************************************/
CORBA::Short SPowerTest::get_index(CORBA::Environment &jj)
{       std::cout << "Returning index to client " << clindx+1 <<"..."<< std::endl;
        return (CORBA::Short)(++clindx);
}/* End of function get_index of classs SpowerTest
                */
/************************************************************************/
/* Class        SPowerTest
                                                  */
/* Function         add_client              */
/* Description  Adds a client to the end of the client list     */
/* Date             8/16/2000               */
/* Modification                                                  */
/* References                                                    */
/*-----------------------------------------------------------------------*/
```

```
/* PreCond                  ior is a valid IOR reference          */
/*-------------------------------------------------------------------------*/
/* PostCond                 client with IOR will be at end of client list   */
/*************************************************************************/
CORBA::Boolean SPowerTest::add_client(const char * ior,  /* An stringified IOR reference*/  const char *
name,ORBA::Environment &jj) /* A name for the client        */
{       VehicleModule::vehicleClient_var client;  /* The cleints server
                */
  cnode *temp,  /* Temporarily holds the new client                                  */
                *end;  /* Holds the last cnode in client list                                  */
        int argv=1;

        std::cout << "Starting AddClient ..."<< std::endl;
        //   try{
    //Initialize the ORB
                char *jnk[1] = {"SServer"};
                std::cout << "Initializing AddClient ORB..."<< std::endl;
                CORBA::ORB_var orb = CORBA::ORB_init(argv, jnk, "");

    //obtain the root context
                std::cout << "Narrowing AddClient ORB..."<< std::endl;
                CORBA::Object_var obj = orb->string_to_object(ior);
                std::cout << "IOR Converted..." << std::endl;
                std::cout << "IOR is: " << ior << std::endl;
                client = VehicleModule::vehicleClient::_narrow(obj);
                std::cout << "Client Gotten..." << std::endl;

        /* If the client list is empty                                                  */
    if(clist == NULL)
    {   /* Set new node to clist                                                        */
                        std::cout << "CList Empty Adding New CLient..."<< std::endl;
                        clist = new cnode();

//                      clist->next = NULL;

                        clist->vclient = client;
                        strcpy(clist->name, name);
    }
    else
    {   /* Else put new client at end of list
        */
                        std::cout << "Clist Not empty Adding New Client..."<< std::endl;
                  temp = new cnode();               /* Create a new node
                        */

                        temp->vclient = client;   /* Add client to new node
        */

                        strcpy(temp->name, name);       /* Add client name to new node
                */


                        end = clist;                            /* Set end to first node in list
                */
```

```
                        while(end->next != NULL) {    /* walk through list
                        */
                                cout << "Got here 5\n";
                                end = end->next;
                        }

    end->next = temp;          /* Add temp to end of list                                    */
  }/* End if                                    */

  // Determine what type of client it is...
        std::cout << "Checking Client Type..."<< std::endl;
  if(name[0] == 'v')
  {  /* Vehicle Client
                                */
                clnum++;
    /* If client is first vehicle to connect return true
      */
                if(clnum == 1) return true;
  }
  else
  {  /* IETM Client                                                                    */
                ienum++;
                /* If client is first IETM to connect return true
      */
    if(ienum == 1) return true;
  }/* End if                                                          */
  // }
  // catch (CORBA::Exception &)
  //    {          std::cerr << "CORBA exception raised!" << std::endl;
  //    }

      /* Else client isn't first of its type to conect, return false
      */
      std::cout << "Exiting AddClient..."<< std::endl;
 return true;
}/* End of function add_client of class SPowerTest                                    */
/*************************************************************************/
/* Class          SPowerTest
                                        */
/* Function               update_clients
                                        */
/* Description    Performs callback operation on all clients in client list.          */
/* Date                   8/16/2000
                                          */
/* Modification
                                          */
/* References
                                          */
/*-------------------------------------------------------------------------*/
/* PreCond                none
                                          */
/*-------------------------------------------------------------------------*/
/* PostCond               Client RPM will be updated
                                        */
/*************************************************************************/
void SPowerTest::update_clients()
```

```
{
   int i = 0;        /* A simple counter
                     */
   cnode *tback = NULL; /* A temporary client pointer                                    */
   cnode *temp = clist; /* A temporary client pointer                          */
         /* Loop through the client list                                              */
         std::cout << "Starting UpdateClients()..."<< std::endl;
   while(temp != NULL)
    {  //try{
                              /* try to update the client                             */
                              std::cout << "Updating Client "<< i << " with RPM=" << (frame->getRPM())
<< std::endl;
         temp->vclient->set_rpm((short)(frame->getRPM()));
              //       }
              //catch (CORBA::Exception &)
              //                   {          /* If the client didn't respond, remove it from client list
                        */
              //                   if(temp->name[0] == 'v')
        //    --clnum;
              // else
        //    --ienum;
              //
        //if(tback != NULL)
              //                              {
              //  tback->next = temp->next;
        //    temp = NULL;
        //    temp = tback;
              //                   }
              //       else
              //       {
              // clist = temp->next;
              // temp = NULL;
              // temp = clist;
              // continue;
       // }/* End if                                                                   */
   // }/* End try/catch                                                               */
      tback = temp;
      temp = temp->next;
      ++i;
  }/* End while                                                                      */
  std::cout << "Exiting UpdateClients()..."<< std::endl;
}/* End of function update_clients of class SPOwerTest                          */

/*************************************************************************/
/* Class             SPowerTest
                                                */
/* Function                    update_clients (parameter)
                                    */
/* Description       Updates the RPM and parameters for all Clients
          */
/* Date                        8/16/2000                                          */
/* Modification                                                                   */
/* References        update_clients()                                             */
/*-------------------------------------------------------------------------*/
/* PreCond                     none
                                                      */
```

```
/*-------------------------------------------------------------------------*/
/* PostCond                 all clients will have current RPM
                            */
/*                                  IETM clients will have updated parameter          */
/*************************************************************************/
void SPowerTest::update_clients(char p[]) /* Paramater
                  */
{
   int i = 0;       /* A simple counter
                  */
   cnode *tback = NULL; /* A temporary client node
                  */
   cnode *temp = clist; /* A temporary client node
                  */

         std::cout << "Starting UpdateClients(Char)..."<< std::endl;
   while(temp != NULL)
{  //try{
                           /* update the clients                                                    */
                           std::cout << "Updating Client..."<< i << std::endl;
       (temp->vclient)->set_rpm((short)frame->getRPM());

                           /* If the cleint is an IETM, update the test parameter
        */
                           if(temp->name[0] == 'I')
        (temp->vclient)->set_testparm(p);
                           // }
                           // catch (CORBA::Exception &)
                           //                {       /* A client can not be contacted and will be
                                                    removed          */
                           // if(temp->name[0] == 'v')
                           // --clnum;
                            // else
                           //       --ienum;

                           //         if(tback != NULL)
                           //         {
                           //       tback->next = temp->next;
                           //  temp = NULL;
                           //   temp = tback;
                           // }
                           // else
                           //         {
                           //  clist = temp->next;
                           //  temp = NULL;
                           //  temp = clist;
                           //   continue; /*Continue while
                                        */
                           //   }/* End if                                                          */
                           //  }/* End try/catch                                                   */

    tback = temp;
    temp = temp->next;
    ++i;
  }/* End while                                                                  */
         std::cout << "Exiting UpdateClients(char)..."<< std::endl;
```

```
}/* End of function update_clients of class SPowerTest                    */

/*************************************************************************/
/* Class          SPowerTest                                             */
/* Function              update_clients (acceleration, percentage)       */
/* Description    Updates RPM for all clients and returns acceleration and percentage */
/*                                        to IETM clients                */
/* Date                  8/16/2000                                                */
/* Modification                                                          */
/* References     update_clients()                                       */
/*-----------------------------------------------------------------------*/
/* PreCond               none                                                     */
/*-----------------------------------------------------------------------*/
/* PostCond              All clients will have an updated RPM            */
/*        IETM clients will have an updated acceleration & percentage result     */
/*************************************************************************/
void SPowerTest::update_clients(float a,   /* Acceleration from test_power */
                                float pcf) /* Percentage from test_power      */
{
   int i = 0;                           /* A simple counter
                                        */
   cnode *tback = NULL; /* A temporary client node
                        */
   cnode *temp = clist; /* A temporary client node
                        */

        std::cout << "Starting UpdateClients(acc, pcf)..."<< std::endl;
   while(temp != NULL)
    { // try{
                        /* Update the client                                */
                        std::cout << "Updating Client..."<< i << std::endl;
          (temp->vclient)->set_rpm((short)frame->getRPM());
                /* If it's an IETM update the results
                */
          if(temp->name[0] == 'I')
                        (temp->vclient)->results(a, pcf);
        // }
        // catch (CORBA::Exception &)
        //      {  /* The client could not be contacted and will be removed
        */
        // if(temp->name[0] == 'v')
        //    --clnum;
        // else
        //    --ienum;
        //
        // if(tback != NULL)
        //                 {
    //    tback->next = temp->next;
    //    temp = NULL;
    //    temp = tback;
        //                      }
    //  else
        //                      {
    //    clist = temp->next;
        //    temp = NULL;
        //    temp = clist;
```

```
        //    continue;
      //  }/* End if                                                          */
        //          }/* End try/catch                                        */
        //
      tback = temp;
      temp = temp->next;
      ++i;
    }/* End while                                                           */
        std::cout << "Exiting UpdateClients(acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest                     */

/*********************************************************************/
/* Class            SPowerTest
                                                    */
/* Function update_clients (parameter, acceleration, percentage)          */
/* Description    Updates RPM for all clients and returns acceleration and percentage*/
/*        and parameter to IETM clients                                   */
/* Date                    8/16/2000                                      */
/* Modification                                                           */
/* References     update_clients()                                        */
/*-------------------------------------------------------------------------*/
/* PreCond                 none
                                                          */
/*-------------------------------------------------------------------------*/
/* PostCond                All clients will have an updated RPM           */
/* IETM clients will have an updated parameter, acceleration &            */
/*                                    percentage result                   */
/*********************************************************************/
void SPowerTest::update_clients(char p[],  /* Parameter from test_power */
        float a,  /* Acceleration from test_pwer                  */
        float pcf) /* Percentage from test_power                  */
{
   int i = 0;       /* A simple counter
                    */
   cnode *tback = NULL; /* A temporary client node                  */
   cnode *temp = clist; /* A temporary client node                  */

        std::cout << "Starting UpdateClients(Char, acc, pcf)..."<< std::endl;
   while(temp != NULL)
    {  //try{
        /* Update the client                       */
                std::cout << "Updating Client..."<< i << std::endl;
         (temp->vclient)->set_rpm((short)frame->getRPM());
                /* If it's an IETM client update parameter and results
       */
        if(temp->name[0] == 'I')
        {       (temp->vclient)->set_testparm(p);
           (temp->vclient)->results(a,pcf);
                }/* End if                                                */
        // }
        // catch (CORBA::Exception &)
        // {      /* The client did not respond so remove it from the list
       */
        // if(temp->name[0] == 'v')
        //    --clnum;
        // else
```

```
            //    --ienum;
            //
            // if(tback != NULL){
            //    tback->next = temp->next;
            //    temp = NULL;
            //    temp = tback;
            // }
            // else{
            //    clist = temp->next;
            //    temp = NULL;
            //    temp = clist;
            //    continue;
            // }/* End if                                                    */
            // }/* End try/catch                                             */

        tback = temp;
        temp = temp->next;
        ++i;
    }/* End while                                                           */
        std::cout << "Exiting UpdateClients(char, acc, pcf)..."<< std::endl;
}/* End of function update_clients of class SPowerTest
                */
```

vehicleframe.cpp:

```
* File              VehicleFrame.cpp                                       */
/* Class  VehicleFrame                                                    */
/*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include "powertypes.h"
#include <sys/types.h>
#include <sys/timeb.h>
#include <string.h>

/*************************************************************************/
/* Class             VehicleFrame                                        */
/* Function                  VehicleFrame                                */
/* Description       Constructs the vehicle frame, initializing all values */
/* Date                      8/16/2000                                   */
/* Modification                                                          */
/* References                                                            */
/*-----------------------------------------------------------------------*/
/* PreCond                   none                                        */
/*-----------------------------------------------------------------------*/
/* PostCond                  All internal values will be initialized     */
/*************************************************************************/
VehicleFrame::VehicleFrame()
{       setvid(123433);
        setRPM(0);
        setAccelerator(0);
        setStartStop(false);
        setLabelOnOff("Off");
        setjLabel4(0);
}/* End of VehicleFrame Constructor                                      */

/*************************************************************************/
/* Class             VehicleFrame                                        */
```

```
/* Function              VehicleFrame(vehicle ID)                              */
/* Description    Constructs the vehicle frame, initializing all values and setting */
/*        vehicle ID to input parameter                                        */
/* Date                  8/16/2000                                             */
/* Modification                                                                */
/* References     VehicleFrame()                                               */
/*-----------------------------------------------------------------------------*/
/* PreCond               v is a valid vehicle ID                               */
/*-----------------------------------------------------------------------------*/
/* PostCond              vehicle ID will be set to input parameter             */
/*                              All internal values will be initialized        */
/*****************************************************************************/
VehicleFrame::VehicleFrame(int v)
{       setvid(v);
        setRPM(0);
        setAccelerator(0);
        setStartStop(false);
        setLabelOnOff("Off");
        setjLabel4(0);
}/* End of VehicleFrame Constructor w/VID                                      */

/*****************************************************************************/
/* Class          VehicleFrame                                                 */
/* Function              Set Functions                                         */
/* Description    Set the individual values within the VehicleFrame            */
/* Date                  8/16/2000                                             */
/* Modification                                                                */
/* References                                                                  */
/*-----------------------------------------------------------------------------*/
/* PreCond               none                                                  */
/*-----------------------------------------------------------------------------*/
/* PostCond              Specific value will be set                            */
/*****************************************************************************/
void VehicleFrame::setAccelerator(int acc)
{       Accelerator = acc;
}/* End of function setAccelerator from Class VehicleFrame                     */

void VehicleFrame::setjLabel4(int rpm)
{       char temp[11];    /* Temporarily holds the new label text             */

//strcpy(temp, "RPM = ");
        //_itoa(rpm, temp,10);
        //strcpy(jLabel4, temp);
}/* End of function setjLabel4 from Class VehicleFrame                         */

void VehicleFrame::setLabelOnOff(char onoff[])
{       strcpy(LabelOnOff, onoff);
}/* End of function setjLabel4 from Class VehicleFrame                         */

void VehicleFrame::setRPM(int rpm)
{       RPM = rpm;
}/* End of function setRPM from classs VehicleFrame                            */

void VehicleFrame::setStartStop(bool running)
{       StartStop = running;
}/* End of funciton setStartStop form class VehicleFrame                       */
```

```
void VehicleFrame::setvid(int v)
{       vid = v;
}/* End of function setvid from class VehicleFrame                                */

/*************************************************************************/
/* Class          VehicleFrame
                                                */
/* Function               Get Functions
                                                */
/* Description    Get the individual values within the VehicleFrame & return to caller*/
/* Date                   8/16/2000
                                                */
/* Modification
                                                */
/* References
                                                */
/*-------------------------------------------------------------------------*/
/* PreCond                none
                                                */
/*-------------------------------------------------------------------------*/
/* PostCond               Specific value will be returned
                                */
/*************************************************************************/
int VehicleFrame::getRPM()
{       return RPM;
}/* End of function getRPM from class VehicleFrame                           */

int VehicleFrame::getvid()
{       return vid;
}/* End of function getvid from class VehicleFrame                           */

int VehicleFrame::getAccelerator()
{       return Accelerator;
}/* End of function getvid from class VehicleFrame                           */

bool VehicleFrame::getStartStop()
{       return StartStop;
}/* End of function getvid from class VehicleFrame                           */
```

cnode.cpp:

```
/*************************************************************************/
/* File           cnode.cpp                                                */
/* Class cnode                                                             */
/*************************************************************************/
#include <stdio.h>
#include <string.h>
#include "vehiclemoduleS.h"
#include "powertypes.h"

/*************************************************************************/
/* Class          cnode                                                    */
/* Function               cnode                                            */
/* Description    cnode Constructor initializes new cnode to null          */
/* Date                   8/16/2000                                        */
/* Modification                                                            */
/* References                                                              */
```